Modelling and Analysis for Cyber-Physical Systems: An SMT-based approach

Dung, Phan Anh

Publication date:
2015

Document Version
Publisher's PDF, also known as Version of record

Modelling and Analysis for Cyber-Physical Systems: An SMT-based approach

Anh-Dung Phan

Kongens Lyngby 2015
PHD-2015-373

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, Building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk PHD-2015-373

Summary

This thesis focuses on high-level modelling and analysis of Cyber-Physical Systems (CPS). The rationale is that: since modelling and analysis phases are closely related to the design phase, having better modelling and analysis techniques would tremendously increase quality of designs. Moreover, better designs have positive impacts on the product quality, development time and price, etc.

We developed tools, theories and techniques that make use of *SMT solving* as a back-end engine for analysis and employ *Duration Calculus* as a front-end technology for modelling. The proposed techniques have been validated via a few interesting case studies.

In particular, a combination of techniques including *reduction to SMT solving*, *novel simplification* for quantified formulas in Linear Integer Arithmetic and *multicore parallelism* has been used to make Duration Calculus feasible for practical use. Duration Calculus has shown its potential as a domain specific language in a Smart Meter case study. Moreover, *counting semantics* has proven useful in connection with tool-based support for Duration Calculus.

To extend SMT techniques towards better support for analysis of CPS, we proposed algorithms for handling quantifier alternations and implemented SMT-based optimization procedures. The optimization procedures, available as an extension to Z3 SMT solver, have been instrumental to provide solutions for our case studies in a natural way.

Resumé

I denne afhandling er fokus på højniveaumodellering og analyse af Cyber-Physical systemer. Rationalet bag dette fokus er at modellerings- og analysefaserne er tæt forbundet til designfasen, og derfor vil forbedrede modellerings- og analyseteknikker være vigtige faktorer omkring etablering af design af høj kvalitet. Ydermere vil bedre designs have stor positiv indflydelse på, for eksempel, udviklingstiden, kvaliteten af det endelige produkt og produktets pris.

I dette projekt er der udviklet værktøjer, teorier og teknikker der gør brug af *SMT solving* som en analyse backend og som benytter *Duration Calculus* som en frontend teknologi til modellering. De etablerede teknikker er validerede ved brug af nogle få case studier.

Kombinationen af teknikker indbefatter blandt andet *reduktioner til SMT solving* og *nye forenklinger* af lineær aritmetiske formler over heltal med kvantorer, og *multicore parallelisme* er benyttet i en stræben på at bringe anvendelse af Duration Calculus nærmere en praktiske brug. Duration Calculus har vist sit potentiale som domæne specifikt sprog i et Smart-Meter case studie. Derudover har en *counting semantics* vist sin betydning omkring værktøjsbaseret understøttelse af Duration Calculus.

Udvidelser til SMT teknikker, der er rettet mod øget analyseunderstøttelse for CPS, inkluderer algoritmer til at håndterer formler med dybe kvantoralterneringer, samt SMT-baserede optimeringsprocedurer. Disse optimeringsprocedurer, tilgængelige som en udvidelse af Z3 SMT solver, har været instrumentale omkring opnåelse af resultater for vores case studier på en naturlig facon.

Preface

This thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark i.e. DTU Compute, in partial fulfilment of the requirements for acquiring the degree of Doctor of Philosophy.

The thesis deals with modelling and analysis of Cyber-Physical Systems. The main focus is to develop theories, tools and techniques that assist developers in the design process of Cyber-Physical Systems.

The thesis is self-contained and revolves around the work done in a number of publications written during the period 2012-2015.

The PhD project has been funded by the IDEA4CPS project granted by the Danish Research Foundation for Basic Research (DNRF86-10). The work has been inspired by the joint effort amongst IDEA4CPS partners and the fruitful collaboration with researchers at Microsoft Research and Verimag.

Lyngby, June 2015

Anh-Dung Phan

Acknowledgements

The first person I would like to thank is my main supervisor, Michael R. Hansen. We met each other four years ago during the course of my Master thesis. I have known one thing back then: he would be the best PhD supervisor one could dream of. Michael has been always available as a source of both inspiration and guidance over the years. I really appreciate his support and encouragement through the highs and lows of my PhD study. Without him, this work would not be possible to attain.

I also would like to thank my co-supervisor, Jan Madsen. He has provided great insights in the area of Cyber-Physical Systems and real-world application. Jan has always had a different way of looking at research problems and come up with interesting ideas and suggestions. Thanks to him, it is possible for us to keep balances between the practical and the theoretical side of the work.

Part of this work was carried out during my internship at Microsoft Research Redmond. I would like to thank my collaborator and mentor, Nikolaj Bjørner at Microsoft Research, for his guidance. I met Nikolaj three years ago while attending his course on SMT solving at DTU Compute. His introduction to SMT solving has been a turning point of my PhD study where SMT solving has served as a foundation for analysis methods of our work. I am grateful to have had a chance to visit Microsoft and learned how formal methods are being used in the industry.

Thanks to all the fellow PhD students at the Embedded Systems Engineering

section at DTU Compute. You are great colleagues with whom I have had interesting discussions inside and outside work. Especially I would like to thank my office mates: Florian Brandner, Luke Thomas Herbert, Qasim Mahmood Rajpoot and Othoman Elaswad for the fun time and inspiring talks we had together. I express the appreciation to my friends and colleagues, Nhut Nguyen and Linh Hong Vu. Thank you for reminding me to maintain a work life balance and to keep myself calm during difficult times.

Finally, the greatest appreciations go to my family: my mother, my father and my brother. They have been supportive for any path I chose. My son, Dan, who has taught me to be the most responsible, and who lights up the dark during my tough times. And the love of my life, my wife, Xuan. She has always been supportive and understanding. You make it possible for me to be where I am.

List of papers

The dissertation is based on the following publications, which are referred to in the text by Paper A-F.

A. M. R. Hansen, A. D. Phan, and A. W. Brekling,
"A practical approach to model checking Duration Calculus using Presburger Arithmetic", *Annals of Mathematics and Artificial Intelligence*, vol. 71, no. 1-3, pp. 251–278, 2014.

B. A. D. Phan and M. R. Hansen,
"An approach to multicore parallelism using functional programming: A case study based on Presburger Arithmetic," *Journal of Logic and Algebraic Programming*, vol. 84, no. 1, pp. 2–18, 2015.

C. A. D. Phan, N. Bjørner, and D. Monniaux,
"Anatomy of alternating quantifier satisfiability (work in progress)", in *10th International Workshop on Satisfiability Modulo Theories, SMT 2012, Manchester, UK,* pp. 120–130, EasyChair, 2012.

D. A. D. Phan, M. R. Hansen, and J. Madsen,
"EHRA: Specification and Analysis of Energy-Harvesting Wireless Sensor Networks", in *Specification, Algebra, and Software - Essays Dedicated to Kokichi Futatsugi*, vol. 8373 of *Lecture Notes in Computer Science*, pp. 520–540, Springer, 2014.

E. N. Bjørner and A. D. Phan,
"νZ - Maximal Satisfaction with Z3", in *6th International Symposium on Symbolic Computation in Software Science, SCSS 2014, Gammarth, La Marsa, Tunisia*, pp. 1–9, EasyChair, 2014.

F. N. Bjørner, A. D. Phan, and L. Fleckenstein,
"νZ - An Optimizing SMT Solver", in *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015*, vol. 9035 of *Lecture Notes in Computer Science*, pp. 194–199, Springer, 2015.

Contents

Chapter 1

Introduction

For quite some time, Cyber-Physical Systems (CPS) have emerged as a field of research closely related to Embedded Systems. The term Cyber Physical Systems is often used to emphasize an integration of computation components with physical processes as stated in [43]: "In CPS, embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa. The design of such systems, therefore, requires understanding the joint dynamics of computers, software, networks, and physical processes." In this chapter, we will discuss Cyber-Physical Systems and their characteristics, and position the focus of this work in the CPS area.

1.1 What are Cyber-Physical Systems?

For many years, Embedded Systems have played an important role in our daily life. For example, we often use cars which are embedded systems comprising many components. Even common traffic lights contain embedded computers running with limited resources and fulfilling real-time requirements. Embedded Systems

get more and more complex and many tasks performed by them today are too critical to tolerate failures.

Two well-known examples of failures that result in either huge financial costs or life threatening to many people are: (1) In 1996, Ariane 5, a European heavy lift launch vehicle, was self-destructed a few seconds into its first test flight. It turned out that a unit conversion bug in the control software has caused the $370M failure. (2) Between 1985 and 1987, Therac-25, a radiation therapy machine produced by Atomic Energy of Canada Limited, was involved in massive radiation overdose to six patients. The reason was a race condition between concurrent tasks.

These incidents show how safety and reliability are of extreme importance to embedded systems design. Therefore, more formal design methods are crucial in designing these systems. In this thesis, one of the recurring themes is to apply formal methods to detect flaws in designs and eliminate them as early as possible.

Recently there has been a growing interest in an emerging category of embedded systems called Cyber-Physical Systems. The cyber part stands for computational components that integrate with physical processes via network fabrics. A distinctive property of CPS is their emphasis on *interaction* and *coordination* between *computational* and *physical* components. The idea is that embedded systems no longer operate in isolation; they often cooperate via networks and coordinate to form a bigger and more autonomous system. For instance, if we allow cars to communicate with traffic lights and self-coordinate for more efficient traffic flows, this forms an intelligent transport system, i.e. an exemplary CPS. More efficient traffic flows give huge boosts in terms of time saving for people in societies. However, intelligent transport systems are complex and error-prone. Imagine if communication signals between cars and traffic lights got delayed, it could cause catastrophic failures where cars crashed and the whole transport system broke down. It becomes more and more important to be able to manage complexity and mitigate failures of such critical systems. The phenomenon inspires a strive for better methodologies and tools to construct CPS.

Unmanned Aerial Vehicles: A motivating example

In December 2013, Amazon announced their Prime Air plan to use drones for speedy delivery. The idea is to use drone-like "octocopters" to deliver lightweight orders in 30-minute timeframe. The octocopters operate in 10-mile radius and carry items weighed less than 2 kg. Although Prime Air has not yet reach

Figure 1.1: A detailed look at an Amazon's octocopter

commercial use, it has steered a lot of discussion regarding the use of Unmanned Aerial Vehicles (UAV) in a large scale [17].

Amazon's vision is that octocopters pick up packages in warehouses and deliver them at customers' doorsteps quickly and safely. Each octocopter is planned for economical delivery, and it needs coordination with other octocopters for route planning and collision avoidance. Although the scenario is very simple, it is notoriously tricky to make it efficient and robust in reality.

The first issue is *safety*. Amazon would like to ensure that drones will not cause injuries, e.g. by dropping goods. To be practical, these flying machines have to be able to operate in a wide range of conditions. But what should be done if a delivery is expected in bad weather? A solution could be that the drones should be equipped with reliable trackers so that they are able to adjust order departures based on weather conditions.

Another issue is *interaction*. An octocopter needs to navigate through unknown territories. It is not straightforward to land reliably on inhomogeneous or even dynamic customer sites. The drones should interact with the environment, capture relevant data and make use of computer vision techniques to find an ideal landing location.

There is also an issue with *coordination*. A drone might deliver multiple packages in a single mission. But disruption in the environment might cause its mission plan to change. In an ideal case, multiple drones should communicate with each other and coordinate where each of them should go next. This is a general problem in *mission planning*, that occurs in package delivery, monitoring and

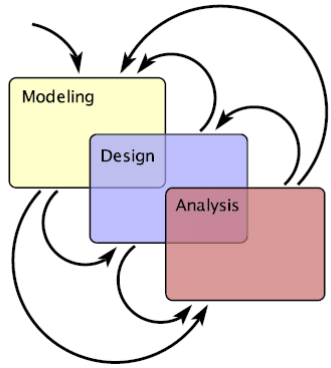


Figure 1.2: An iterative process of model-based design of Cyber-Physical Systems [43]

other use cases.

It is exciting to see applications of Cyber-Physical Systems in daily life. As we can observe from the Prime Air example, it is difficult to build CPS in a sustainable manner. Therefore, it is of extreme importance to have proper feedback in the design process so that faulty design decisions could be ruled out as early as possible. We employ a model-based design methodology. This methodology fits nicely with the emphasis on formal methods as well. Model-based design is discussed in more detail in the next section.

Model-based Design of Cyber-Physical Systems

In order to design Cyber-Physical Systems using a model-based approach, we often have to go through an iterative process with three phases: modelling, design and analysis as shown in Fig. 1.2. Modelling gives a deeper understanding of the problem by creating a specification of what a system should do. This step often builds a mathematical model where the behaviors are specified in a clear and concise manner using a formal modelling language. The design process puts together all components to refine the model into a concrete implementation. This is often done by adding more information to the model and specifying non-functional properties of the system. In the analysis phase, we break the system down to have detailed looks at its different components and their interactions.

From a formal methods' perspective, this step often exploits formal analysis of the models. Three phases – modelling, design and analysis – stand for *what*, *how* and *why* a system does what it does, respectively. We often go back and forth between the phases, gain more insights about the system and refine it along the way. The phases also give multiple levels of abstractions of the system. This is important since one can focus on relevant details at the current level and easily recognize design flaws by using a right abstraction.

The focus of this dissertation is on high-level modelling and analysis techniques. The perspective is by doing qualitative and quantitative modelling and analysis, we get deeper insights on system designs. The acquired knowledge is crucial for making Cyber-Physical Systems safer, more robust and more sustainable. Another goal is to reduce cost of design, development and maintenance in the long term. In the next section, we shall discuss the techniques for modelling and analysis of such systems.

In this thesis, we set our focus on techniques to model and analyze Cyber-Physical Systems. One of the recurring topics in our research is Wireless Sensor Network (WSN). A Wireless Sensor Network is a distributed network of sensor nodes operating in a physical environment. Each node interacts with the environment and collects data for a specific purpose, e.g., environmental monitoring, traffic control, etc. Collected data is gathered at so-called base stations where it is further processed. A long-lasting issue of WSN is to account for unpredictable behaviors of the environment.

If we add energy harvesting capabilities to sensor nodes, their lifetimes now depend on how the environment behave. This adds more complexity to designing WSN because sensor nodes are often designed to sustain long lifetime using low powers. Energy-harvesting-aware routing requires *interaction* and *coordination* between *computational* and *physical* components. Since energy harvesting is unpredictable, sensor nodes need to adapt their behaviors when surrounding environment changes. These *uncertainty* and *complexity* issues make WSN design very challenging and fascinating. By working on this down-to-earth WSN topic, one could understand more about modelling and analysis of WSN and generalize lessons learned to other domains.

1.2 Modelling and Analysis of Cyber-Physical Systems

In this section, different approaches for modelling analysis of CPS are presented. The three categories include automata-based, logic-based and other approaches. While the two former categories are formal in the sense that they use mathematical models of different kinds as a basis for rigorous analysis. The "other approaches" are more heterogeneous; they are based on informal reasoning to gain insights about systems under investigation. There are not always significant differences between these approaches. However, by categorizing them, we are able to explore advantages and drawbacks of each category. The ultimate goal is to use an approach that is most relevant to a given scenario.

1.2.1 Automata-based approaches

Automata, i.e. finite-state machines, are often used to model systems. An automaton comprises states and transitions between states. It provides a convenient and operational modelling formalism via a graphical representation. Complex systems could be addressed by decomposing into small systems which can be represented by independent automata. Then large systems can be modelled by compositions of corresponding automata, where the automata typically are combined in parallel.

One common question is to check whether a model of a system satisfies a given property P. This is so-called *model checking* problem, i.e., verifying correctness properties of finite-state systems [15]. A property P is typically a specification expressed in a form of a formula of a temporal logic. Model checking algorithms then check satisfiability of this formula automatically. Despite being powerful analysis techniques, model checking algorithms face a big challenge due to combinatorial complexity of state spaces i.e. the *state explosion* problem.

There is a lot of research in the area of automata theoretic approaches for CPS. This section does not mean to provide an exhaustive survey of the approaches. Instead we focus on two prominent formalisms including hybrid automata and timed automata and discuss their applications in more detail.

A hybrid automaton is a model comprising finite-state machine and continuously evolving variables [36], where two kind of state changes are considered: (1)

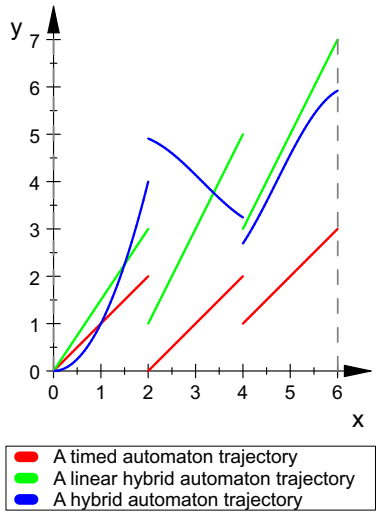


Figure 1.3: Trajectories of different classes of hybrid automata

discrete transitions, happening instantaneously, which are suitable to model discrete systems and (2) continuous flows which are based on time elapses. Continuous variables are depicted by ordinary differential equations. Hybrid automata are well-suited to model CPS since they can describe inherent interaction between *physical* and *computational* components. Due to the presence of continuous variables, hybrid automata are infinite-state systems which classic model checking algorithms are not directly applicable for.

Linear Hybrid Automata (LHA) is a special class of hybrid automata, that allows any differential equation of the form $\dot{x} = k$ where k is constant. It has been received a lot of research attention; a main reason is that LHA is simple yet powerful enough to model a lot of complex systems. Reachability analysis on LHA is undecidable while bounded reachability is still decidable [9]. HyTech, a symbolic model checker for LHA, is able to perform parametric analysis to deduce design parameters for given systems satisfying some temporal requirements [37].

Timed automata, a modelling formalism proposed by Alur et al. [1], is a class of hybrid automata where continuous variables are clocks whose derivatives are constants of the form of $\dot{x} = 1$. Fig. 1.3 shows different trajectories of various classes of hybrid automata. As one can see, the trajectory of a general hybrid automaton has some irregularities that pose serious challenges for automatic

analysis. Timed automata is an attractive formalism since their model-checking problem is decidable [1]. Consequently there are many efficient tools for analyzing timed-automata-based systems e.g. Kronos [20, 73] and Uppaal [4, 42].

Model checking algorithms start with a region automaton construction. The idea is to construct equivalence classes of states in terms of regions. These regions form finite-state automata, which are subjected to classic model checking algorithms.

Revisiting our motivating example on UAV, Seibel et al. has used Hybrid Automata as the formalism for UAV mission planning [63]. The focus of the work is detailed modelling of aircrafts' internal as well as external conditions i.e. the underlying environment. For instance, fuel consumption is specified by a linear hybrid automaton which has each of its states as an estimation of real consumption. The environment is characterized by a set of inequalities in which non-linear regions are approximated by a union of linear regions. This hierarchical approach leads to a very comprehensive set of hybrid automata, which are composed together in parallel to form a complex system. A flight plan is then deduced from a reachability analysis of the whole system. An interesting aspect of using Hybrid Automata is that one can model interaction between computation and physical components through combination of continuous dynamics and discrete state changes. Moreover, the environment is also taken into account, which makes the model realistic.

1.2.2 Logic-based approaches

The separation between automata-based and logic-based approaches is not always crystal clear. A model checker often uses automata for detailed modelling and logics for specification thanks to their expressiveness. In this section, an approach is said to be logic-based if solving these logical formulas is the main ingredient of the approach.

Advantages of logic-based approaches are their conciseness and expressiveness in modelling systems and properties. These strengths allow users to encode complex systems and provide useful abstractions in understanding such systems. A trade-off of expressiveness is high complexity of decision procedures for such logics if such procedures exist.

There are different kinds of logic-based approaches. We mention here a few common ones.

- *Interactive theorem proving* Using these approaches, users state explicit goals in a proof assistant, and gradually prove them based on hints provided by the proof assistant. Due to the *interactive* nature, the problem-solving process might be tedious and error-prone for complex problems. A few examples of such proof assistants are Isabelle [51] and PVS [66].
- *Automatic theorem proving* A specific logic fragment is used to model systems and a corresponding automatic prover deduces results from input formulas. In the area of CPS, differential dynamic logic ($\mathsf{d}\mathcal{L}$), introduced by Platzer, is an expressive formalism to specify and verify hybrid systems [56]. The logic can be used to specify properties in first-order formulas combined with hybrid programs. The properties can be verified using the deductive verification tool for hybrid systems, KeYmaera. There are a number of case studies using $\mathsf{d}\mathcal{L}$ to verify complex CPS such as European Train Control System and aircraft collision avoidance manoeuvres [58, 57]. An advantage of $\mathsf{d}\mathcal{L}$ is its advanced support for compositional modelling of complex systems.
- *Reduction-based techniques* These methods rely on reductions from modelling formalisms to first-order logics. In the end one may still use automatic theorem proving in order to solve logic formulas. However, reductions have many intricacies that need special treatments. This section focuses on reduction-based approaches. We demonstrate a few methods of this category later in more detail.

Bounded model checking (BMC) is one of the representative techniques for reduction-based approaches. Different from classical model checking, BMC unrolls the finite-state machine until a fixed number of steps k. The algorithm verifies whether a property is violated in up to k steps by checking satisfiability of a logical formula.

BMC is a commonly-used technique to remedy the *state explosion* problem. The approach is well-suited for finding bugs/counterexamples. Its drawback is non-exhaustiveness; it is only able to find possible violations up to a value k. Bounded model checking for Linear Hybrid Automata has been addressed by the work of Audemard et al. [2] for example. The work relies on Boolean satisfiability i.e. SAT solving and employs mathematical reasoning over the reals. Experiments were conducted on Secondary Power System, a concrete case study in the CPS area. Experimental results showed that the approach is more scalable than that of HyTech.

On a related note, Gao et al. also encodes bounded reachability of hybrid systems

using Satisfiability Modulo Theories (SMT) solvers [33]. SMT solvers can be understood as SAT solvers plus reasoning capabilities w.r.t. some background theories. The authors consider reachability analysis within a number of discrete transitions. To deal with non-linear nature of hybrid systems, $\delta - complete$ decision procedures that ensure systems to be safe or unsafe under certain thresholds are proposed. Their novel SMT solver **dReal** is able to handle problems with thousands of variables [32].

Coming back to our motivating UAV example, Humphrel et al. proposed to use Linear Temporal Logic (LTL) as a specification language for synthesizing UAV mission plans [38]. Mission plans often include UAVs' movements and their paths on specific regions of road networks. The richness of LTL is really important in expressing interesting mission specifications. One can encode common UAVs' properties e.g. coverage, sequencing and avoidance in LTL in a concise manner. The idea is that UAVs' road networks are discretized to concrete points using maximum and minimum speed possible. The whole system of multiple UAVs is modelled as a transition system using these points as states. These transition systems are rather simple; the focus is in the coordination between the UAVs, which generates comprehensive mission specifications. Mission specifications are written as temporal properties of these locations; then model checking techniques are used to synthesize a feasible mission plan. The underlying model checker is in turn based on logical formalisms e.g. the properties are reduced to Boolean formulas to be solved by a SAT solver.

Reduction-based techniques are being used frequently throughout the dissertation. One example is to solve model checking problem of Duration Calculus, an interval temporal logic, that allows succinct ways to express real-time properties [12]. The main idea of our approach is to reduce the model checking problem to a formula in Linear Integer Arithmetic and to use a powerful backend to solve formulas in that logic. This problem involving Duration Calculus inspires several directions of our work and constitutes the core of the thesis. We shall describe the approach at a deeper level in Chapter 3.

1.2.3 Other approaches

Simulink, developed by MathWorks, is an environment for modelling and simulating systems with time-dependent behaviors. The environment is based on Block Diagrams comprising input and output signals. The Block Diagrams can be easily composed to create more complex systems. Simulink is widely used

in the domain of control designs and it starts being used for modelling CPS [48]. It has many strengths e.g. code generation support for multiple platforms and advanced simulation capabilities. Furthermore, Simulink Design Verifier provides excellent tools to detect design flaws that cannot be found by simulation [71].

Modelica, an open standard developed by Object Management Group, provides an object-oriented modelling language for CPS [31]. Models in Modelica are often represented graphically that give users opportunities to reason about the models informally. A collection of crafted models is provided to support modelling of systems of systems in a large scale. The emphasis of Modelica is not formal analysis but flexibility in modelling; it also provides powerful simulation capabilities. For example, Modelica's simulation of controller timing is based on TrueTime simulator [60].

While automata-based and logic-based approaches often suffer from scalability problems, simulation-based approaches are applicable for large systems. This makes simulation appealing to detailed modelling and analysis of concrete systems. One method is to use multiple levels of abstractions i.e. approximated high-level models for formal analysis and precise system-level models for simulation purposes. This method facilitates the use of complementary approaches which have their own strengths to offer.

A method that is in between formal and simulation-based approaches is *statistical model checking*. The premise of statistical model checking is scalability by not exhausting state spaces. Moreover, it still has guarantees up to certain probability similar to what model checking offers. Statistical model checking has excellent tool support such as Uppaal SMC [10] for timed automata.

While Simulink and Modelica are excellent tools to examine and simulate concrete systems, there are some gaps in composing large-scale systems to be filled. For the purpose, one can use a standardized language for model-based systems engineering e.g. SysML and AADL [35, 25]. The advantage of using such a language is that it gives more holistic views on systems of systems and provides tools to examine specific components and their interactions.

1.3 Goals of the thesis

In the previous section, various approaches for modelling and analysis of CPS have been introduced. We believe each approach should be used on areas of

their best strengths. Therefore, combined methods are often employed where we can inject relevant approaches at appropriate places. Our intention is to develop theories, tools and techniques for modelling and analysis of CPS and apply those to concrete case studies. The goals of this thesis can be summarized as follows:

- Develop theories, tools and techniques for Duration Calculus, that are geared towards high-level modelling and analysis of CPS. As mentioned in Section 1.2, Duration Calculus is used as a *front-end technology* to solve problems of CPS nature. Moreover, in making DC's model checking algorithms more efficient, we are inspired to investigate various techniques to make Duration Calculus suitable for practical use.
- Develop theories, tools and techniques for SMT solving, that are geared towards analysis of CPS. SMT-based analysis could be used as a *back-end technology* for various problems. Along with the use of reduction to SMT solving, we examine the use of SMT optimization. Problems of CPS nature often require quantitative analysis in a form of optimization.
- Develop modelling and analysis techniques for Wireless Sensor Networks with special focus on energy harvesting. In particular, we analyze global behaviors of such networks and use concrete WSN topologies as case studies to gain insights on the problems at hand and on the analysis techniques.

1.4 Structure of the thesis

The remaining chapters are organized as follows.

- Chapter 2 gives an introduction on SMT solving and Duration Calculus, the main modelling and analysis frameworks we choose as a basic of the work. The chapter provides basic knowledge to understand the techniques presented in the thesis.
- Chapter 3 summarizes our publications and their main contributions. The chapter consists of detailed summaries for Paper A-F.
- Chapter 4 describes two cases in the CPS domain: the first one on Smart Meter of home appliances and the second one on simple WSN in a chain topology. The former case is relatively simple; it focuses only on *coordination* between appliances. The latter case takes into account

both *coordination* and *interaction* between sensor nodes. These concrete examples are examined using the techniques presented in Chapter 3. The idea is to revisit relevant techniques and discuss their applicability in appropriate circumstances.

- Chapter 5 concludes the thesis with a brief summary and a discussion of future work.

Chapter 2

Background

In Chapter 1, we set out the goals to employ Duration Calculus and SMT solving as the front-end and back-end technologies, respectively, to address CPS problems. This chapter provides the readers with background knowledge on SMT solving and Duration Calculus in order to understand concrete contributions and their relationships in the next chapter.

We start with a quick glance of SMT solving, that provides analysis techniques being used throughout the thesis. The introduction shows what SMT solving is and how it is used in practice. The next section gives a brief introduction to Duration Calculus, that was proposed in connection with specification and analysis of hybrid and embedded systems. The use of Duration Calculus as a modelling formalism will be described in more detail in Chapter 3 and Chapter 4.

2.1 Introduction to SMT solving

What is SMT solving?

For a Boolean formula φ^b, there exist algorithms for checking satisfiability of φ^b. Such an algorithm is called *SAT solver*. However, formulas arise in practice do not only contain Boolean fragment; they often require reasoning based on certain theories. Taking the following formula φ as an example

$$\varphi \equiv (2x + 1 = 2y) \wedge b$$

where $b \in \mathbb{B}$. If x and y are on the domain of reals, φ is satisfiable. However, if the domain of integers is chosen, no satisfying assignment exists for φ.

For a first-order logic formula φ, a key question is:

"Is formula φ satisfiable modulo theory $\mathcal{T}$?"

If the answer is yes, the formula is said to be $\mathcal{T}$-satisfiable. The question is central for so-called Satisfiability Modulo Theories (SMT) problems. Procedures that solve SMT problems are called *SMT solvers*.

Theory $\mathcal{T}$ can be understood as a signature defining domains, functions and relations of the theory. For instance, Linear Real Arithmetic (LRA) has its signature to be given by domain $\mathbb{R}$ of reals, and relations $\geq, <, \leq, >, =$. The operations consist of addition $(+)$, subtraction $(-)$, and multiplication $(\times)$ by a constant $r \in \mathbb{R}$. Efficient decision procedures for LRA exist; they are implemented as an integral part of most modern SMT solvers [24]. If changing the domain from $\mathbb{R}$ to $\mathbb{Z}$, one obtains theory of Linear Integer Arithmetic (LIA). LIA and LRA are two theories being used frequently in this thesis as we shall see in Chapter 3.

Another theory, Equality and Uninterpreted Functions (EUF), constitutes the core of SMT solvers. The signature contains uninterpreted functions and relations. An important property is *congruence rule*: if $t = s$ then $f(t) = f(s)$. EUF is the core theory of SMT because most of theories are translated to EUF, for which efficient decision procedures exist. The readers are referred to the congruence closure algorithm [23, 3] for more thorough treatments.

To illustrate these theories, let us take a look at the following example which has a mixture of LRA and EUF theory:

$$\varphi \equiv (x + 2 = y) \wedge (f(3) \neq f(y - x + 1)) \tag{2.1}$$

If we substitute y by $x+2$ and perform a simplification, we arrive at the following formula:

$$\varphi \equiv (x + 2 = y) \wedge (f(3) \neq f(3))$$

The formula is inconsistent i.e. unsatisfiable w.r.t. the EUF theory. As the example demonstrated, combination of theories is an important advantage of SMT solving. SMT provides an expressive modelling formalism where multiple background theories can be leveraged. This also means *high-level abstractions* for users where many problems are too tedious to express in pure SAT. Another strength is that not every theory can be reduced to SAT.

SMT solving: A simple example

Input: Ground formula φ in theory $\mathcal{T}$
Output: unsat or sat with a satisfying assignment in $\mathcal{T}$
$\varphi^b \leftarrow fol2prop(\varphi)$
while $sat_{\mathbb{B}}(\varphi^b)$ **do**
  $\beta^b \leftarrow pick\_assignment(\varphi^b)$
  $(\rho,\ \pi) \leftarrow sat_{\mathcal{T}}(prop2fol(\beta^b))$
  if ρ *is sat* **then**
    return *sat*
  end
  $\varphi^b \leftarrow \varphi^b \wedge \neg fol2prop(\pi)$
end
return *unsat*

Algorithm 1: A core algorithm for SMT solving [22]

The core algorithm of an SMT solver is given in Algorithm 1. First, the first-order formula φ is converted to a Boolean abstraction φ^b on fresh propositional variables by using function $fol2prop$. Then we enumerate satisfying assignments of the Boolean abstraction and check for satisfiability of φ in theory $\mathcal{T}$. The procedure $sat_{\mathbb{B}}$ is a *SAT solver*, that answers whether the formula φ^b is satisfiable. If it is *sat*, the SAT solver provides a model containing all Boolean atoms, β^b. A SAT solver is often based on the DPLL algorithm [19], but we do not focus on SAT solving in this chapter. The interested readers are referred to [5] for background information.

The Boolean model is translated back to first-order atoms in $\mathcal{T}$ using function $prop2fol$. The procedure $sat_{\mathcal{T}}$ i.e. *Theory solver* is used to check for consistency

w.r.t. theory $\mathcal{T}$. The theory solver returns two results: (1) ρ is an outcome, being either *sat* or *unsat* (2) π is an optional value which returns an unsat core i.e. a set of conflicting literals when ρ is *unsat*. To reiterate, if ρ is *unsat*, the theory solver provides evidence π in a form of theory conflicts, which are translated back to Boolean domain (using $fol2prop$) in order to exclude this inadequate SAT model from next SAT solvers' results. The procedure in Algorithm 1 continues until $sat_{\mathcal{T}}$ returns *sat* or all Boolean satisfying assignments have been enumerated.

We use the following example to illustrate SMT solving with LIA theory:

$$\varphi \equiv (x \geq 0) \wedge (y = x + 1) \wedge (y > 2 \vee y < 1)$$

A few steps of solving this formula are given in Fig. 2.1 with annotated labels based on Algorithm 1. Here are a number of remarks:

- The Boolean abstraction φ^b is created by simply declaring new propositional variables $p_1,\ p_2,\ p_3$ and p_4 for first-order literals.
- φ^b has a model in $\mathbb{B}$ where $p_1,\ p_2,\ \neg p_3,\ p_4$ are true. However, there does not exist a corresponding model in $\mathcal{T}$.
- $sat_{\mathcal{T}}$ gives a result stating that $x \geq 0,\ y < 1, y = x + 1$ are conflicting literals which constitute an unsat core.
- $fol2prop$ translates negation of the unsat core to Boolean formula $\neg p_1 \vee \neg p_2 \vee \neg p_4$. This extra information is used in next SAT iteration to exclude current model.

The key takeaway is that the exchange of information between SAT solver and a theory solver is done in a modular way. In a nutshell, an SMT solver comprises of a SAT and a theory solver that exchange atoms with each other.

Architecture of an SMT solver

The main components of an SMT solver are shown in Fig. 2.2. The architecture depicted here is heavily influenced by that of Z3 SMT solver [21]. However, many of the components can also be found in other SMT solvers.

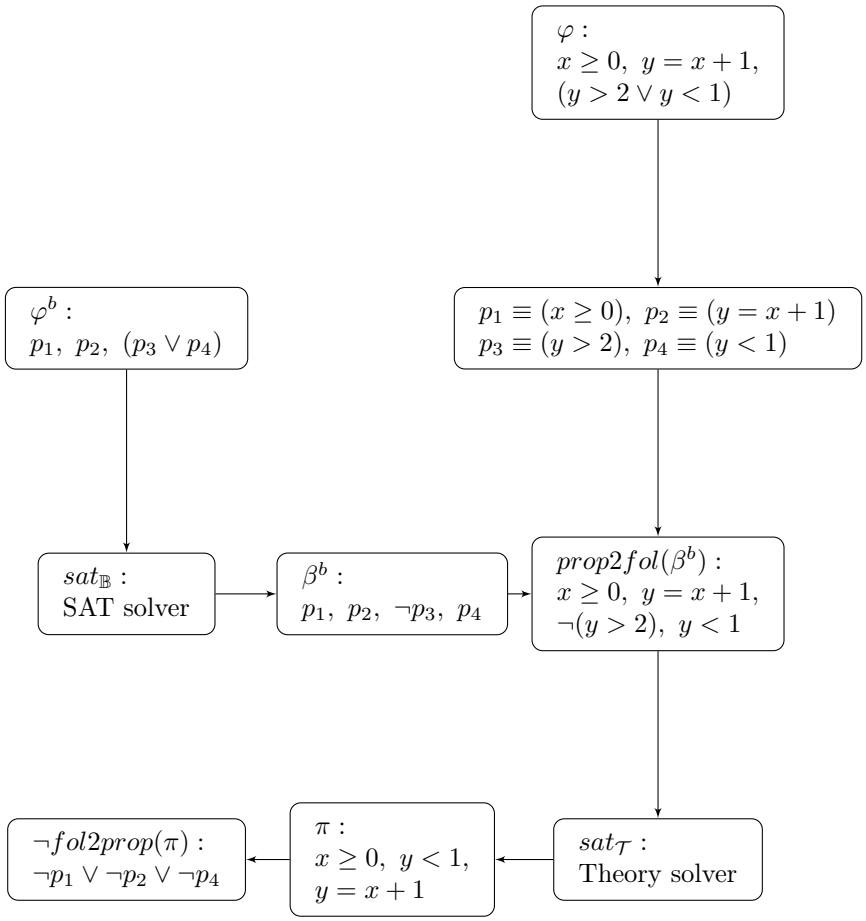


Figure 2.1: An example trace of SMT solving

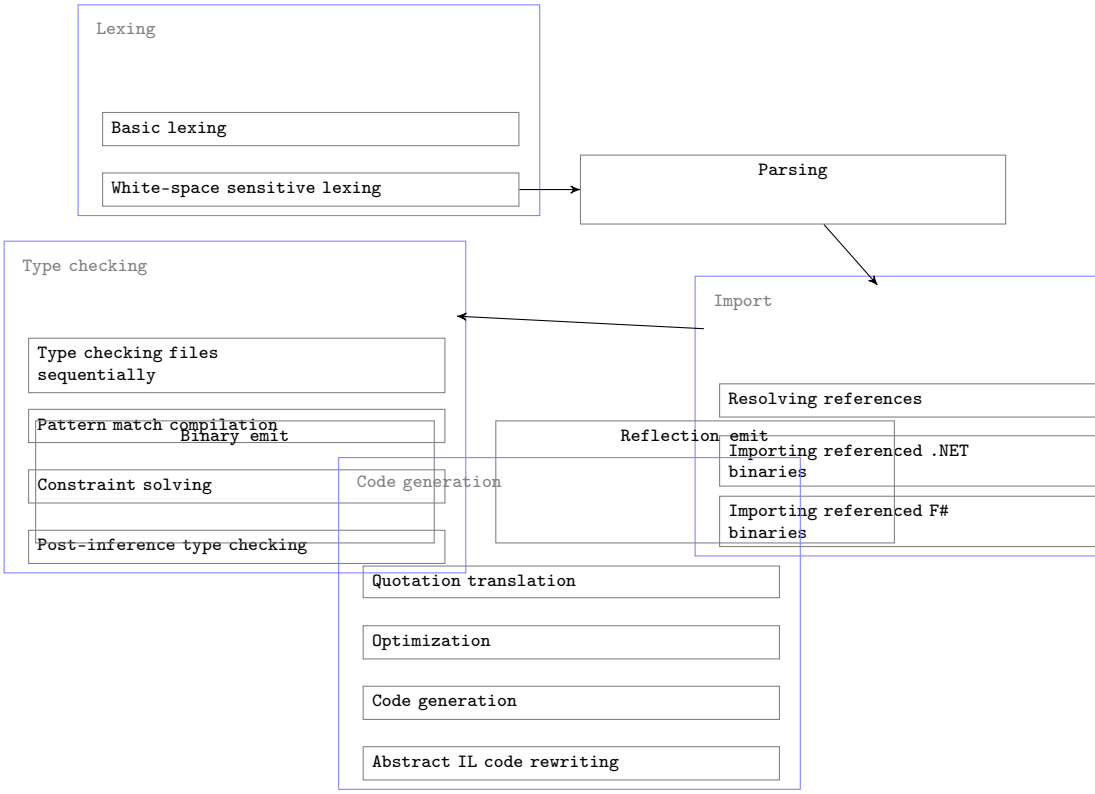


Figure 2.2: An architecture of an SMT solver

- **Simplifier**: This component performs simplifications that exploit the structure of input formulas. As shown in Equation 2.1, many formulas consist of redundant assertions and effective simplification is enough to give definite results in certain cases. In Paper A, we discuss simplification techniques that are akin with this component of an SMT solver.
- **Compiler**: The component compiles abstract syntax trees of formulas into different data structures suitable for being processed by core theory solvers.
- **Congruence closure core**: This is the core SMT solver that understands SMT formulas in EUF theory. A general SMT problem is translated to a SMT one with only equalities to be handled by this solver.
- **SAT solver**: This is a typical SAT solver that sends literal assignments back to the core theory solver for SMT solving w.r.t. equalities.
- **Satellite theory solvers**: Most of theories supported by SMT are implemented as satellite theory solvers. As demonstrated in Fig. 2.1, all the theory-specific bits are abstracted away from input formulas. The core SMT problem with equalities is solved by the core SMT solver. Whenever a satisfying assignment was constructed, the theory solver would check it for consistency w.r.t. the background theory.

From the architecture above, one can see that a benefit of implementing a theory solver as part of an SMT solver is the tight integration with SMT core. It means that there are useful information e.g. equality assignments and atoms exchanged between solvers to make use of. Another advantage is to avoid doing redundant work. An SMT solver only passes a small chunk of a formula to corresponding SAT solver at a time. If an SMT problem is reduced to SAT upfront, it may suffer significant blowup even before the solving phase.

Beside theory of linear arithmetic (including LIA, LRA, etc.), we are also interested in theory of quantifiers. For a first-order logic formula with quantifiers, a quantifier-elimination procedure is an algorithm that finds a quantifier-free equivalence of such formula. There are efficient algorithms for quantifier elimination; we shall go into more detail in Paper C.

Z3 is a state-of-the-art SMT solver from Microsoft Research [21]. The SMT solver supports satisfiability of many of theories out of the box. Paper E and Paper F present an extension of Z3 SMT solver, that provides optimization capabilities on top of satisfiability. The optimization procedures are also implemented as satellite theory solvers. For a first-order logic formula and an optimization goal of some

kind, one needs to find a satisfying assignment of the formula that optimizes the given goal. In a typical case, optimization solvers interact with the core SMT solver to retrieve a solution for the core SMT problem. However, there are some cases where input formulas can be compiled to pure SAT problems. The optimization theory solvers then invoke the core SAT solver directly for better efficiency.

SMT-based problem solving: N-Queens example

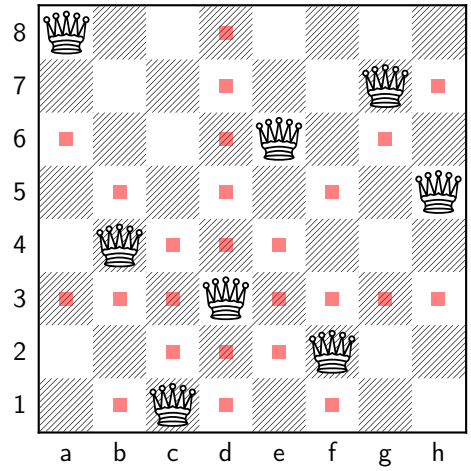


Figure 2.3: A 8×8 chessboard with a feasible solution

Consider the following classic problem: **placing n queens on an $n \times n$ chessboard so that no two queens attack each other**. A simple SMT formulation using Linear Integer Arithmetic is established as follows:

- Let $\mathrm{Q}_i \in \mathbb{Z}$ be a non-negative integer variable for $1 \leq i \leq n$.
- $\mathrm{Q}_i = k$ denotes that a queen is placed at position $(i,\ k)$, i.e. column i and row k.
- Every queen is placed in the board:

$$\mathrm{Q}_i \geq 1 \wedge \mathrm{Q}_i \leq n \text{ where } 1 \leq i \leq n$$

```
(assert (distinct (- Q1 Q2) -1 1))
(assert (distinct (- Q1 Q3) -2 2))
(assert (distinct (- Q1 Q4) -3 3))
(assert (distinct (- Q1 Q5) -4 4))
(assert (distinct (- Q1 Q6) -5 5))
(assert (distinct (- Q1 Q7) -6 6))
(assert (distinct (- Q1 Q8) -7 7))
...
(assert (distinct (- Q5 Q6) -1 1))
(assert (distinct (- Q5 Q7) -2 2))
(assert (distinct (- Q5 Q8) -3 2))
(assert (distinct (- Q6 Q7) -1 1))
(assert (distinct (- Q6 Q8) -2 2))
(assert (distinct (- Q7 Q8) -1 1))
```

Figure 2.4: Diagonal constraints for N-Queens problems with N=8

- Each queen is in a distinct row:

$$\mathrm{Q}_i \neq \mathrm{Q}_j \text{ where } 1 \leq i < j \leq n$$

- No diagonal can contain two queens:

$$\mathrm{Q}_i - \mathrm{Q}_j \neq i - j \wedge \mathrm{Q}_i - \mathrm{Q}_j \neq j - i \text{ where } 1 \leq i < j \leq n$$

In Fig. 2.3, a solution for 8×8 chessboard is shown. For example, since there is a queen at position $d3$, no other cell in the same row, column or backward/forward diagonals of $d3$ can contain a queen.

Fig. 2.4 illustrates the diagonal constraints above in SMT-LIB syntax. SMT-LIB is the de-factor language for SMT solving, that allows expressing SMT problems in a concise and clear manner. As shown in Fig. 2.4, translation from logical formulation to SMT-LIB syntax is straightforward. Throughout the thesis, we use the logical notation, the translation to SMT-LIB is assumed to be trivial.

Notice that the formalization to Linear Integer Arithmetic is one of many possible encodings for the N-Queens problem. For instance, one can exploit the fact that positions comprise of bounded natural numbers and make use of a fixed-width bit-vector encoding. An advantage of SMT solving (compared to SAT counterpart) is their high-level abstraction where one can focus on expressing problems in

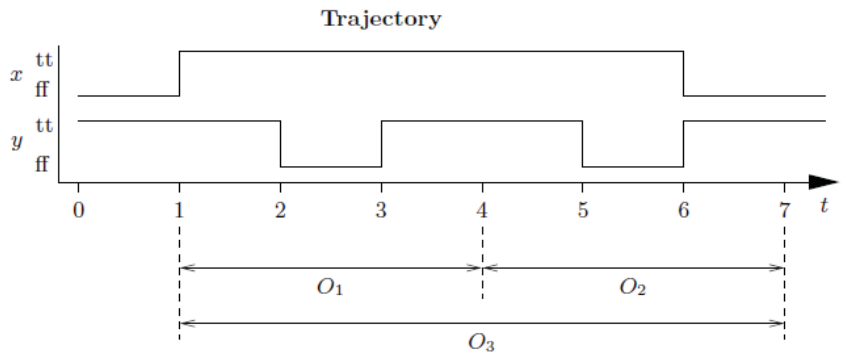


Figure 2.5: A trajectory on observation intervals

natural ways. Most of the heavy-lifting tasks are done by theory solvers under the hood.

2.2 Introduction to Duration Calculus

Interval Temporal Logic (ITL), introduced by Moszkowski [47], is a real-time logic that allows quantitative interval properties to be expressed in a natural way. A formula ϕ in ITL is constructed via following rules:

$$\phi \ ::= \ \theta_1 = \theta_2 \mid \neg\phi \mid \phi \vee \psi \mid \phi \frown \psi \mid ...$$

where *chop* ($\frown$) is the only modality of the logic. A formula $\phi \frown \psi$ holds in an interval [b, e] if the interval can be partitioned into two consecutive intervals [b, m] and [m, e] so that ϕ holds in [b, m] and ψ holds in [m, e].

Duration Calculus (DC), another real-time logic, extends ITL with a notion of duration $\int_b^e S(t)dt$ [12, 11]. This extension allows succinct formulation of quantitative properties concerning durations of complex properties.

Fig. 2.5 describes an exemplary trajectory with multiple intervals. One can observe that DC formula $(\int x \geq 3) \wedge (\int y < 3)$ holds on interval [1, 4] and $\int y - 2\int(x \wedge \neg y) = 0$ holds on interval [4, 7] respectively. By the definition of chop, DC formula $((\int x \geq 3) \wedge (\int y < 3))\frown(\int y - 2\int(x \wedge \neg y) = 0)$ holds on interval [1, 7].

Undecidability of decision problems is a big challenge in dealing with DC. The decision problem of DC is undecidable unless there are severe restrictions imposed on formulas and/or models [34, 28, 29]. Consider a discrete-time DC fragment whose syntax is defined as follows:

$$S ::= 0 \mid 1 \mid P \mid \neg S \mid S_1 \vee S_2$$

$$\phi ::= \top \mid \Sigma_{i \in \Omega} c_i \int S_i \bowtie k \mid \neg\phi \mid \phi \wedge \psi \mid \phi^\frown\psi$$

where P is in the set of *state variables*, $k,\ c_i \in Z$ and $\bowtie \in \{<,\ \leq,\ =,\ \geq,\ >\}$. The satisfiability problem of the above fragment is undecidable already unless the duration notion is limited to the form $\int S \bowtie k$.

A similar issue occurs for the model checking problem of DC. We consider the model checking problem $K \models \phi$, where K is a finite automaton and ϕ is a DC formula which consists of atomic formulas of the form $\Sigma_i c_i \int S_i \bowtie C, \bowtie \in \{<, \leq, =, \geq, >\}$. This problem is undecidable for both continuous-time and discrete-time semantics. In Paper A, we examine an approximation-based semantics for DC to arrive at a decidable DC fragment.

A Duration Calculus example: Recovery from Damages

This example is based on the one given in Paper A. We describe here a system with damage recovery. The system may suffer from damages due to uncontrollable events. Once damage has occurred, the system requires some recovery actions to return to its normal operation. Some damages are more severe than others, so stronger recovery is required in some cases.

A model of the system is depicted in Fig. 2.6. There are two state variables D_1 and D_2 for damages, and three state variables R_i, where $1 \leq i \leq 3$, for recoveries. The normal operation mode is at vertex 0. When damages occur, the system is in vertices 1 or 3 depending on severity of damages. Recovery strengths have to exceed certain threshold before the system can come back to normal operations.

Sufficiency of the recoveries can be formulated by a DC formula where an accumulated weighted duration of the recovery states exceeds that of the damage states. As seen from Equation 2.2, the notion of duration provides a concise and expressive way of modelling system properties. Here constants $k_i,$ where $1 \leq i \leq$

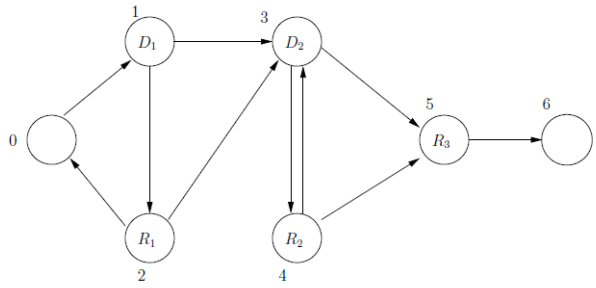


Figure 2.6: Example: Recovery from Damages

3, and c_j, where $1 \leq j \leq 2$, express strengths of damages and recoveries.

$$\Box(\int \mathrm{R}_3 \geq 1 \implies \Sigma_{i=1}^{3} k_i \int \mathrm{R}_i \geq \Sigma_{j=1}^{2} c_j \int \mathrm{D}_j) \tag{2.2}$$

In Chapter 4, the same fragment of DC is used as a modelling formalism for a problem of Cyber-Physical nature.

2.3 Summary

In this chapter, we have given brief introductions to SMT solving and Duration Calculus. While the former provides core analysis techniques used throughout the thesis, the latter is a suitable formalism for the Cyber-Physical problems of our interest. By presenting background introduction, one hopes that the readers have enough context for understanding individual contributions in Chapter 3.

Chapter 3

Contributions

Paper A solves the model checking algorithm of Duration Calculus using an approximation-based semantics via a reduction to Presburger Arithmetic. The main contributions are the use of *counting semantics* for efficient encoding and *Guarded Normal Form* for practical simplification. The model checking algorithm also motivates the research in Paper B and Paper C.

In Paper B, a thorough parallelization scheme was proposed using a language-based cost model and concrete practical development guidelines. Promising speedups were recorded for Cooper's algorithm and the exact shadow in Omega Test, the two prominent decision procedures for Presburger Arithmetic. These performance gains contribute to the overall improvement of the model checking algorithm. The parallelization scheme has wide applicability e.g. for other advanced tree algorithms.

In Paper C, we approach the problem in a totally different angle. Noticing that the Presburger formulas generated from the model checking problem have deep nesting of alternating quantifiers, a generic algorithm for solving alternating quantified formulas was proposed. Multiple efficient instantiations of the generic algorithm were implemented in Z3 SMT solver. Experimental results show that the alternating quantifier satisfiability algorithm gives a significant performance

gain in solving quantified formulas of alternating nature.

In Paper D, we come back to Wireless Sensor Networks, our central research topic in the CPS domain. The setting is to investigate three different routing protocols and analyze their influences on global behaviors of the network. The main contributions are (1) a formal modelling framework where semantics of sensor nodes and routing protocols can be described in a precise way (2) several global properties are analyzed on the basis of a simulation-based technique.

In subsequent papers, we revisit SMT solving in the context of optimization. The motivation is that optimization is a recurring pattern in the CPS domain. Paper E and Paper F present two aspects of νZ, the optimization framework as part of Z3 SMT solver. While the former has its focus on algorithms and implementation techniques, the latter emphasizes the modelling side i.e. how the framework can be used for problem solving. As a logical consequence, the two papers will be presented in the same section and motivated via a common example.

3.1 A practical approach to model checking Duration Calculus using Presburger Arithmetic

The model checking problem

Duration Calculus is a suitable formulation to model Cyber-Physical Systems. One can express quantitative properties using the notion of accumulated durations of states in Duration Calculus. This gives expressiveness with the cost of increasing complexity of decision problems. Even worse, many fragments of Duration Calculus are undecidable unless there are severely constrained assumptions [28, 67]. For example, a decision procedure for a Duration Calculus with quantifiers on Boolean states is implemented in the tool DCVALID [54]. However, this Duration Calculus fragment is too limited to express interesting quantitative properties.

We are interested in the model checking problem $K \models \phi$, where K is a Kripke structure and ϕ is a Duration Calculus formula which consists of atomic formulas of the form $\Sigma_i c_i \int S_i \bowtie C$, $\bowtie \in \{<, \leq, =, \geq, >\}$. This is an expressive fragment of DC where the model checking problem can easily be undecidable. For that reason, we consider an approximation-based *counting semantics* to achieve decidability [30]. The counting semantics calculates visiting frequencies of vertices based on

Parikh image of traces. We employ a similar technique as described in [64] but apply it on vertices instead of flows.

Consider a simple Kripke structure K:

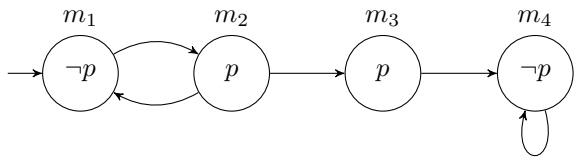


Figure 3.1: A simple Kripke structure

and a DC formula:

$$K \models \ell < 4 \Rightarrow \int p < 3$$

The model checking problem can be translated to satisfiability of the following formula

$$\forall \bar{m}.\ (\text{cons}_{14} \Rightarrow \Sigma_{i=1}^{4} m_i - 1 < 4 \Rightarrow m_2 + m_3 < 3)$$

We denote m_i and e_{ij} as visiting frequencies of vertex m_i and edge e_{ij} (from m_i to m_j) respectively. Here cons_{14} is a *consistency predicate* that ensures a consistent run between vertex 1 and 4. The idea of consistent runs is that flows of incoming edges and outgoing edges should be the same as frequencies of the vertices.

$$\begin{aligned}
&\exists e_{12}, e_{21}, e_{23}, e_{34}, e_{44}. \\
&\qquad e_{21} + 1 = m_1 \wedge e_{12} = m_1 \\
&\wedge \quad e_{12} = m_2 \wedge e_{21} + e_{23} = m_2 \\
&\wedge \quad e_{23} = m_3 \wedge e_{34} = m_3 \\
&\wedge \quad e_{34} + e_{44} = m_4 \wedge e_{44} + 1 = m_4
\end{aligned}$$

Figure 3.2: The formula $\text{cons}_{14}(\bar{m})$ for the Kripke structure in Fig. 3.1.

The generated formulas look simple at first, but their sizes quickly grow out of hand when chop modality is involved. Here is a sketch of an input formula

$$\forall \bar{m}.\ \text{cons}_{ij}(\bar{m}) \Rightarrow \text{markT}_{ij}(\phi \frown \psi)(\bar{m})$$

where markT_{ij} is a *marking formula* denoting the marking between vertex i and j.

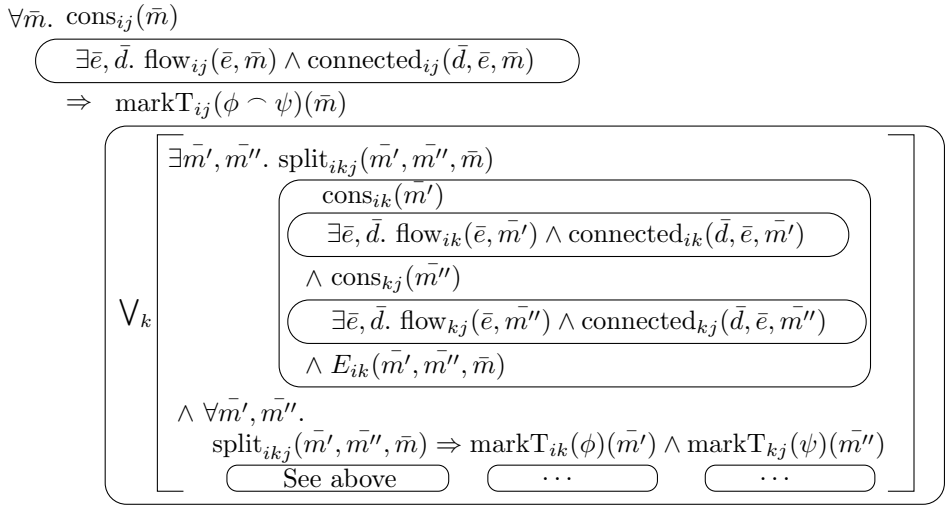


Figure 3.3: A breakdown of a generated formula

In Fig. 3.3, each formula can be expanded as the complex formula in the rounded-corner box right below it. The figure visualizes the recursive structure of the formulas based on the building blocks such as flow_{ij}, $\mathrm{connected}_{ij}$ and split_{ikj}. The generated Presburger formulas consist of complex Boolean formulas with many levels of quantifier alternations. Here cons_{ij} is constituted from flow_{ij} and $\mathrm{connected}_{ij}$. While formula flow_{ij} ensures a consistent flow where visiting frequencies of vertices agree with their flows on incoming and outgoing edges, formula $\mathrm{connected}_{ij}$ models relationships between visiting frequencies of vertices and their distances to start vertex. For each multiset of visiting frequencies, we denote split_{ikj} as a way to construct a run from two consecutive runs having k as the transit vertex.

While we start to be able to use Duration Calculus for modelling, practicality of the approach is still in question. Particularly a double exponential lower bound is established as complexity for checking satisfiability of any Presburger formula [26], and the worst-case running time for a decision procedure for Presburger Arithmetic is known to be triple exponential upper bound [53].

Guarded Normal Form

The generated Presburger formulas may have size exponential to the chop depth of Duration Calculus counterparts. This causes a severe problem in terms of memory usage. In order to shrink Presburger formulas, we introduce Guarded Normal Form (GNF), a non-expansive canonical form where literals are collected into guards to provide contexts for simplifying the rest of the formula.

Guarded formulas consist of *implication guard* $g \Rightarrow \bigvee_i F_i$ and *conjunction guard* $g \wedge \bigwedge_i F_i$ where g is a conjunction of literals. From Equation 3.1 to Equation 3.4, we introduce a series of non-expansive transformations where formulas are flattened and literals are collected into guards. These transformation are to reduce nesting of formulas and accumulate information to central places for simplification.

$$g \Rightarrow \bigvee_i F_i \vee l \quad \Longleftrightarrow \quad \neg l \wedge g \Rightarrow \bigvee_i F_i \tag{3.1}$$

$$g \wedge \bigwedge_i F_i \wedge l \quad \Longleftrightarrow \quad l \wedge g \wedge \bigwedge_i F_i \tag{3.2}$$

$$g \Rightarrow (g' \Rightarrow \bigvee_i F_i) \vee \bigvee_j G_j \quad \Longleftrightarrow \quad g \wedge g' \Rightarrow \bigvee_i F_i \vee \bigvee_j G_j \tag{3.3}$$

$$g \wedge (g' \wedge \bigwedge_i F_i) \wedge \bigwedge_j G_j \quad \Longleftrightarrow \quad g \wedge g' \wedge \bigwedge_i F_i \wedge \bigwedge_j G_j \tag{3.4}$$

Applying the transformations above by left to right order, we eventually arrive at a formula in GNF when no further application is possible.

Guarded Normal Form admits a very nice property. Guards provide contextual information which can be used to simplify the corresponding formulas. Contextual information can even be used in connection with quantified formulas. For example, we have the following reductions for cheap quantifier elimination.

$$\exists x.\ (nx = t \wedge g\ \Rightarrow\ \bigvee_i F_i) \quad \Longleftrightarrow \quad \top \tag{3.5}$$

$$\exists x.\ (nx = t \wedge g\ \wedge\ \bigwedge_i F_i) \quad \Longleftrightarrow \quad n|t \wedge g[t/nx]\ \wedge\ \bigwedge_i F_i[t/nx] \tag{3.6}$$

$$\forall x.\ (nx = t \wedge g\ \wedge\ \bigwedge_i F_i) \quad \Longleftrightarrow \quad \bot \tag{3.7}$$

$$\forall x.\ (nx = t \wedge g\ \Rightarrow\ \bigvee_i F_i) \quad \Longleftrightarrow \quad n|t \wedge g[t/nx]\ \Rightarrow\ \bigvee_i F_i[t/nx] \tag{3.8}$$

Here we assume that $n \in \mathbb{Z} \setminus \{0\}$ and x does not occur in t. These reductions often result in further simplifications. Another interesting observation is that one can easily substitute variables arising from equalities to reduce variable spaces.

Assume that x does not occur in g; one can systematically apply the following equivalences from left to right.

$$(\exists x.\ g \wedge g'\ \wedge\ \bigwedge_i F_i) \quad \Longleftrightarrow \quad g\ \wedge\ (\exists x.\ g'\ \wedge\ \bigwedge_i F_i) \tag{3.9}$$

$$(\exists x.\ g \wedge g'\ \Rightarrow\ \bigvee_i F_i) \quad \Longleftrightarrow \quad g\ \Rightarrow\ (\exists x.\ g'\ \Rightarrow\ \bigvee_i F_i) \tag{3.10}$$

These equivalences happen to be really useful for our generated formulas. The generation process produces a lot of redundant equalities which can be removed for more compact representations. Moreover, most of the quantifiers can be eliminated through the above-mentioned transformations. To give more details on the effectiveness of Guarded Normal Form, we recorded some statistics on formula sizes in Table 3.1.

Lessons learned

As demonstrated in Table 3.1, the use of GNF helps dramatically reduce size of Presburger formulas by exploiting their special structures and providing a cheap way to eliminate quantifiers. Normalized formulas are then being fed into Z3 SMT solver, which has excellent support for Presburger Arithmetic [7]. We

Input	Number of atomic formulas/quantifiers	
	Before GNF	GNF
Consistency predicates (cons_{ij})	33/7	19/3
Flow formulas (flow_{ij})	37/16	8/2
Marking formulas (markT_{ij})	267/103	107/19

Table 3.1: Sizes of generated formulas

also employ the **Simplifier** module inside Z3 to further reduce the formulas. Our experiments demonstrate that both simplification processes are crucial to provide inputs that can be solved by Z3 in a reasonable amount of time.

We investigate two small case studies, one for concatenating a sequence of automata and another for a model of a system that recovers from damages. Although the flavor of Cyber-Physical domain is not so strong, it is a promising start for using Duration Calculus as a modelling formalism. Moreover, the case studies have shown that the approximation nature of counting semantics is good enough to model and analyze useful systems.

The first experiment checks satisfiable and unsatisfiable properties on a Kripke structure composed from a sequence of identical automata. The primitive automaton is given in Fig. 3.1. The goal is to investigate scalability of the approach on big inputs in different situations. Experimental results show that the use of GNF leads to 50% reduction of formula sizes upon Z3 simplification. Many of the original formulas were too big for Z3 to handle. The solving time is also slightly better for GNF formulas.

In the second experiment, we revisit the damage recovery problem introduced in Chapter 2. A model of the problem was presented in Fig. 2.6. To remind the readers, we have two state variables D_1 and D_2 for damages and three ones R_1, R_2 and R_3 for recoveries.

$$\Box \left(\begin{array}{l} \int(\mathrm{D}_1 \vee \mathrm{R}_1) \leq t_{max1} \wedge \int(\mathrm{D}_2 \vee \mathrm{R}_2) \leq t_{max2} \wedge \int \mathrm{R}_3 \geq 1 \\ \qquad\qquad \Longrightarrow \ \Sigma_{i=1}^{3} k_i \int \mathrm{R}_i \geq \Sigma_{j=1}^{2} c_j \int \mathrm{D}_j \end{array} \right) \tag{3.11}$$

Since the focus is parameter synthesis, we use a slightly more complex Duration Calculus formula in Equation 3.11. The first part of the formula (before $\Longrightarrow$) asserts that one should not stay at D_1 or R_1 longer than t_{max1} time instances, should not stay at D_2 or R_2 longer than t_{max2} time instances and should

visit R_3 eventually. The second part of the formula denotes sufficiency of recovery i.e. recovery is successful when accumulated sum of weighted durations on recovery states is larger than that of damaged ones. For given constants k_1, k_2, c_1 and c_2, the goal is to find the smallest coefficient k_3 for the weighted recovery state duration $k_3 \int \mathrm{R}_3$ to satisfy the requirement. The experiment demonstrates an iterative way to synthesize a bound of k_3 given 6 different parameter combinations. The empirical results indicate that GNF is a big plus when being used in connection with Z3 simplification. GNF+Simplication mode takes just a few seconds to synthesize k_3. The default mode without GNF cannot complete the same task within timeout of 5 minutes.

In Paper A, we have solved the high-complexity model checking problem of Duration Calculus by a combination of using contextual simplification (Guarded Normal Form) and a powerful backend solver (Z3). It shows that simplification is very useful in practice if one can exploit the inherent structure of the input formulas. The counting semantics has given us a simple yet powerful tool to tackle modelling problems. In the subsequent papers, we will discuss it as a recurring technique to abstract away irrelevant details in an encoding.

3.2 An approach to multicore parallelism using functional programming: A case study based on Presburger Arithmetic

In the previous section, we proposed to solve the model checking problem of Duration Calculus by a reduction to Presburger Arithmetic. While contextual simplification and SMT solving have given a nice solution, we are interested in better scalability for large formulas. Paper B sets an aim to take advantage of multicore parallelism and functional programming in addressing decision procedures for Presburger Arithmetic.

Multicore processors have become commodity for many years, yet their extra powers have not been fully exploited. Multicore parallelism has been well studied in the context of flat data structures such as arrays, matrices, etc. but there are not many available results on recursive data structures [8]. We study parallelism in solving Presburger formulas that have deeply nested structures, and use functional programming as a powerful tool for the purpose.

While functional programs are relatively easy to parallelize, not many case studies

on parallelization of functional programs on complex tree structures are known. By examining a close-to-home case study, we would like to derive a design methodology and investigate its applicability in a wider scope.

Presburger Arithmetic (PA), also known as Linear Integer Arithmetic (LIA), introduced by Mojzaesz Presburger in 1929, is a first-order theory of integer which accepts + as its only operation. Any Presburger formula is constructed from primitive formulas such as $d \bowtie \Sigma_i c_i x_i$ and $d \mid \Sigma_i c_i x_i$ where $\bowtie \in \{<, \leq, =, \geq, >\}$, d and c_i are constants and x_i are variables.

Beside Boolean combinations, Presburger formulas may contain quantifiers of arbitrary depths. A typical example of representing some amount of money by 3-cent and 5-cent coins is modelled in PA as follows:

$$\forall \mathtt{z}.\ (\mathtt{z} \geq 8 \Rightarrow \exists \mathtt{x}\ \exists \mathtt{y}.\ (3\mathtt{x} + 5\mathtt{y} = \mathtt{z}))$$

Classical decision procedures solve the satisfiability problem by use of quantifier elimination.

Quantifier elimination: Cooper's algorithm

Assuming that ϕ is a formula in Negation Normal Form (NNF), Cooper's algorithm removes quantifiers from ϕ in the inside-out order [7].

$$\exists x.\ \phi \iff \bigvee_{i=1}^{\delta} (\phi[\top/ax < t, \bot/ax > t, i/x] \vee \bigvee_{ax > t \in L} \phi[t+i/ax] \wedge a \mid t+i) \quad (3.12)$$

Here δ is the least common multiple of all divisors d in divisibility constraints $d \mid t$. Moreover, L is the set of all lower-bound constraints of the form $ax > t$ collected from ϕ. We assume that in each round of quantifier elimination, formula ϕ is normalized to contain only lower-bound constraints $ax > t$ and upper-bound constraints $ax < t$. Using the following example with nested quantifiers

$$\exists y\ \exists x.\ 2x + 3y = 7 \wedge x < y$$

we have two systematic transformations

$$\exists y\ \exists x.\ 2x + 3y = 7 \wedge x < y$$
$$\iff \exists y.\ \bigvee_{i=1}^{2} (i < 2 \wedge i > 0 \wedge 5y > 6 + i \wedge 2 \mid 6 - 3y + i)$$
$$\iff \bigvee_{j=1}^{10} \bigvee_{i=1}^{2} (i < 2 \wedge i > 0 \wedge j > 0 \wedge 10 \mid 12 + 2i - 3j \wedge 5 \mid 6 + i + j)$$

The resulting quantifier-free formula consists of multiple disjunctions. This is a big opportunity for parallelization since each round of quantifier elimination generates a number of disjunctions that can be processed in parallel.

Quantifier elimination: the Omega Test

Intuition of the Omega Test results from its geometrical representation and the way the algorithm eliminates a quantifier by projecting it and creating lower-dimension geometrical objects. The algorithm requires input formulas in the Disjunctive Normal Form (DNF) $\bigvee_m \bigwedge_n l_{mn}$ where l_{mn} is a literal.

Let us consider a lower-bound constraint $cx \leq \gamma$ and an upper-bound constraint $bx \geq \beta$. Then we have the following equivalence [59]:

$$\exists x.\ cx \leq \gamma \wedge bx \geq \beta \iff \left(\begin{array}{ll} & c\beta + (c-1)(b-1) \leq b\gamma \\ \vee & c \geq b \wedge c\beta \leq b\gamma \wedge \bigvee_{i=1}^{b-1} (b \mid \beta + i \wedge c(\beta + i) \leq b\gamma) \\ \vee & c < b \wedge c\beta \leq b\gamma \wedge \bigvee_{i=1}^{c-1} (c \mid \gamma - i \wedge b(\gamma - i) \geq c\beta) \end{array} \right) \tag{3.13}$$

Since $\bigwedge_n l_{mn}$ can be transformed to the canonical form $\bigwedge_{i \in L} c_i x \leq \gamma_i \wedge \bigwedge_{j \in U} b_j x \geq \beta_j$ where $c_i > 0$, $b_j > 0$ and L and U are the sets of lower-bound constraints and upper-bound constraints respectively. We have $|L| \cdot |U|$ pairs of corresponding constraints to apply the Omega Test. The constraints $c\beta \leq b\gamma$ and $c\beta + (c-1)(b-1) \leq b\gamma$ are called *real shadow* and *dark shadow* in that order.

When the real shadow is false, the original formula is unsatisfiable. If the dark shadow is true, then the formula is satisfiable. Otherwise, we perform a number of case splittings to explore the formula with concrete values for the given quantifier.

When $c = 1$ or $b = 1$, the real and dark shadow are the same thing called *exact shadow*. In that case, the equivalence becomes:

$$\exists x.\ cx \leq \gamma \wedge bx \geq \beta \iff\ c\beta \leq b\gamma \tag{3.14}$$

This transformation operates on the integers in the same way that Fourier-Motzkin elimination works on the reals.

The parallelization approach

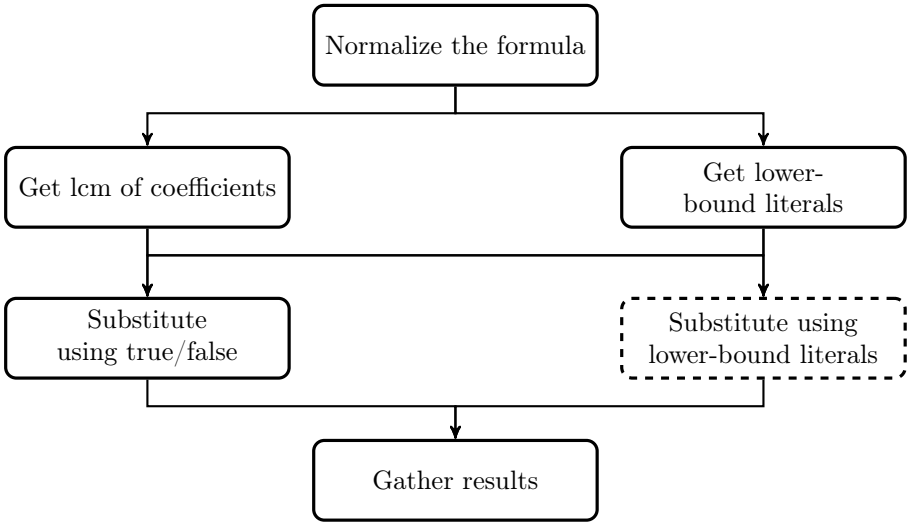
The starting point of our approach is a simple sequential implementation. On that basis, we carefully examine inherent parallelism in the quantifier elimination procedures. The Directed Acyclic Graph (DAG) model of multithreading is employed to analyze parallelism factors inside these algorithms. The model was introduced by Blelloch as a language-based cost model for parallel computing while developing NESL, one of the first functional programming language with implicit parallel constructs. In this cost model, each vertex in a DAG stands for a basic block of computation when an edge describes dependencies amongst computations. Two important concepts were introduced: *work*, denoted by W, representing total amount of computation in the DAG and *span*, denoted by S, showing the slowest sequential path of a computation, i.e. the longest path in the DAG from start vertex to end vertex. Because an algorithm cannot be executed faster than any of its sequential path, speedups on N processors S_N are limited by parallelism factor PF, which is the ratio of work and span:

$$S_N \leq PF = \frac{W}{S}$$

The cost model provides a reliable way to estimate parallelism factors of algorithms. Moreover, the graph representation is an intuitive way to understand and explore parallelization. For example, a quantifier elimination step in Cooper's algorithm is modelled in Fig. 3.4. Let l denote the number of literals in the input formula. In Fig. 3.4, while the dashed box can be completed in $O(|L| \cdot l)$ and any other box has $O(l)$ complexity. This leads us to the parallelism factor:

$$PF = (|L| \cdot l + 4 \cdot l)/(l + l + l) = (|L| + 4)/3$$

To remind the readers, $|L|$ denotes the number of resulting lower-bound constraints in each quantifier elimination. More detailed analysis of the algorithm can be found in Section 4.1 in Paper B.



The dashed box represents $|L|$ substitutions, one for each disjunction.

Figure 3.4: Task graph denoting a round of quantifier alternation

Transformation to parallel execution is clear and concise thanks to the absence of side effects. Although we have validated that parallelism factor is not a problem for Cooper's algorithm and the exact shadow, how much can we achieve in practice?

A number of practical matters can have negative effects on parallelism including *garbage collection*, *cache locality* and *parallelism runtime*. While deriving a parallel implementation, we follow simple guidelines addressing these matters and re-evaluate the implementation in an iterative manner. Here are general observations that are proven to be very helpful for our case studies.

- The parallelism runtime may have big overheads on excessive parallelization. One should control parallelization by setting an appropriate depth of parallelism and figuring out a suitable work partitioning scheme.
- Cache locality is hugely affected by data representation. Our experiments show that an array-based representation often outperforms a list-based representation because the former often allocates contiguous blocks of memory and uses less memory than the latter does. Another trick is to pad arrays by extra elements to make sure that the whole array fits to an

exact number of cache lines.

- Regarding garbage collection, functional programs often allocate short-lived objects which put more pressures on garbage collectors. Reducing memory allocation often improves efficiency of garbage collection and provides better speedups. For example, representing Presburger terms by stack-allocated structs is really effective in eliminating a large amount of short-lived objects.

By following the guidelines mentioned above, we obtained good speedups on an 8-core platform: 4× for Cooper's algorithm and 6× the exact shadow part of the Omega Test. The Omega Test experiments have been performed on Presburger formulas arising from the model checker algorithm such as consistency predicates and marking formulas. This is very encouraging since the exact shadow tests have eliminated quite a lot of quantifiers and the simplification procedure becomes faster thanks to the exploit of multicore parallelism.

Paper B has presented an interesting case study whose inputs come from the Duration Calculus model checking algorithm. We examine the case study in the interplay between functional programming and multicore parallelism. While the results are extremely helpful for the Duration Calculus model checking problem, we believe the approach can also be applied to other advanced tree algorithms. Moreover, by generalizing the exact shadow test, we have a direct result for parallelizing the Fourier-Motzkin elimination method on the reals.

3.3 Anatomy of Alternating Quantifier Satisfiability

For any PA formula of length n, running time of any non-deterministic decision procedure is at least $2^{2^{cn}}$ for some constant $c > 0$ [26]. Moreover, a triply exponential upper bound $2^{2^{2^{cn}}}$ is proven for worst-case running time of Cooper's algorithm [53]. As mentioned in the previous section, Cooper's algorithm and the Omega Test are two common decision procedures for PA although their complexity is inhibitedly high. These decision procedures are also available from SMT solvers such as Yices and Z3.

In Section 3.1, we have investigated decision procedures for Presburger Arithmetic as important tools to solve the model checking problem in Duration Calculus. As demonstrated in Fig. 3.3, an important characteristic of the generated formulas

is their deep nesting of quantifier alternations. Whenever there is an alternation between existential and universal quantification, resulting formulas often blow up in each round of quantifier elimination. This poses a serious challenge in solving quantified formulas efficiently. Our proposed method has helped remedy the situation by compressing the formulas into Guarded Normal Form. However, this method is incomplete and there is a lot of room for further simplifying quantified formulas. Here we demonstrate an alternative approach where we solve quantifier alternations in a single-pass algorithm.

Paper C proposed a generalized algorithm for alternating quantifier satisfiability. There are existing algorithms for Quantified Boolean Formulas, that are specialized for Boolean domain [6, 55]. An algorithm for Linear Real Arithmetic was proposed [45]; the fact that it is tightly coupled with an All-SMT loop prevents it from being applied to other domains.

Inspired by [45], we create a parameterized variant which abstracts the auxiliary procedures. The algorithm can be instantiated by two procedures: one for projection and another for extrapolation. Projection could be understood as partially or fully removing certain quantifiers from input formulas. By using extrapolation, we introduce refinements of input formulas that still satisfy certain conditions.

Extrapolation

We denote $C = \langle A,\ B \rangle$ as an extrapolant of A and B if the following conditions hold:

$$\begin{array}{lll} A \wedge B \text{ is } \texttt{unsat} & \textit{then} & C = false \\ A \wedge B \text{ is } \texttt{sat} & \textit{then} & A \wedge C \text{ is } \texttt{sat}, \\ & & \neg B \wedge C \text{ is } \texttt{unsat} \end{array}$$

As described in Fig. 3.5, one can understand $\langle A,\ B \rangle$ as a refinement of formula B under the context of formula A. There are many choices to implement extrapolation; one procedure is called SMT-TEST. Assume that $\mathcal{L} := \ell_1, \ldots, \ell_n$ is the set of literals in the satisfying assignment for $A \wedge B$. An SMT-TEST extrapolant has an intersection with $\mathcal{L}$ and it is an unsatisfiable set of literals of $\mathcal{L} \wedge \neg B$. The procedure can be implemented efficiently using SMT solvers.

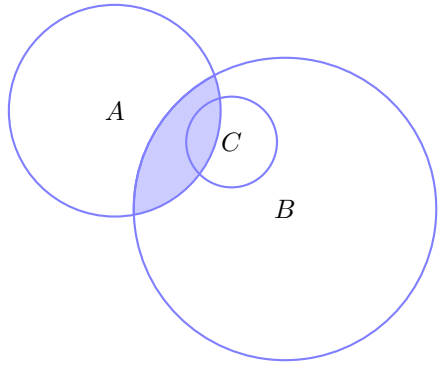


Figure 3.5: An illustration of an extrapolant $C = \langle A,\ B \rangle$

Projection

Intuitively a projection is an image of a quantified formula when we get rid of a quantifier. Assume that C and M are formulas and x is a *free* variable in formula C i.e. $x \in FV(C)$. A projection procedure $\pi x.(C|M)$ computes a refinement of $\exists x\ C$ given the contextual formula M. It means that the following conditions hold:

1. $FV(\pi x.(C|M)) \subseteq FV(C) \setminus \{x\}$
2. $\pi x.(C|M)$ is `sat`
3. $(M \wedge \pi x.(C|M)) \Rightarrow \exists x\ C$

The definition of projection does not dictate how the procedure is implemented. We can freely plug in any quantifier elimination procedure that satisfy the conditions above. Since Linear Integer Arithmetic is the logic fragment of our interest, we can use both Cooper's algorithm and the Omega Test as projections. In Paper C, the implementation makes use of Z3's internal quantifier elimination, which is a mixture of the decision procedures above.

The algorithm

Algorithm 3 (QE) accepts a formula F_n and a vector of quantifiers $\overline{q}$ as its inputs. Each quantifier here stands for an alternation to be removed. The intuition of

QE is that it enumerates all quantifier-free formulas by producing disjoint output formulas and strengthening the input formula for the next call. Algorithm 2 (QT) has a number of inputs including the vector of quantifiers $\overline{q}$, current index i, an input formula C and a vector of contextual formulas $\overline{M}$. The algorithm tries to compute extrapolants of the input formula and a contextual formula M_i at level i repeatedly; it stops when an extrapolant cannot be strengthened any more. Projection is only done on simple extrapolants but we are still able to remove nested quantifier alternations in a single execution.

if $C \wedge \overline{M}_i$ *is* ***unsat*** **then**
  return $(false,\ \overline{M})$
end
if $i\ =\ n$ **then**
  return $(\langle C,\ \overline{M}_i\rangle,\ \overline{M})$
end
$(C',\ \overline{M'}) \leftarrow QT\ (\overline{q},\ i+1,\ \langle C,\ \overline{M}_i\rangle,\ \overline{M})$;
if $C'\ =\ false$ **then**
  return $(\langle C,\ \overline{M}_i\rangle,\ \overline{M'})$
end
$\overline{M''}_k\ \leftarrow\ \overline{M'}_k,\ \forall\ k \neq i$;
$\overline{M''}_i\ \leftarrow\ \overline{M'}_i \wedge\ \neg(\pi\overline{q}_i.(C'|\overline{M'}_i))$;
return $QT\ (\overline{q},\ i,\ C,\ \overline{M''})$;

Algorithm 2: $QT(\overline{q},\ i,\ C,\ \overline{M})$

$\overline{M}_k \leftarrow$ `true`, $\forall\ k < n$;
$\overline{M}_n \leftarrow \overline{F}_n$;
$C \leftarrow$ `false`;
$C' \leftarrow$ `true`;
while $C' \neq$ `false` **do**
  $(C',\ \overline{M}) \leftarrow QT(\overline{q},\ 1,\ \neg C,\ \overline{M})$;
  $C \leftarrow C \vee C'$;
end
return C;

Algorithm 3: $QE(\overline{q},\ \overline{F}_n)$

We use the following small example for illustration:

$$\forall y\ \exists z.\ (z \geq 0 \wedge ((x \geq 0 \wedge y \geq 0) \vee -y - z + 1 \geq 0)) \tag{3.15}$$

The formulas corresponding to Equation 3.15 are:

$$\begin{aligned} F_1 &= \forall y.\ \neg F_2, \\ F_2 &= \forall z.\ \neg F_3, \\ F_3 &= z \geq 0 \wedge ((x \geq 0 \wedge y \geq 0) \vee -y - z + 1 \geq 0) \end{aligned}$$

with a vector of quantifiers $\overline{q} = \{y,\ z\}$.

In Table 3.2 we present a trace of QT algorithm where $x \geq 0$ is the final result of alternating quantifier satisfiability. The formula C propagates information between levels (denoted by i). The idea is that propagating C from level i to level $i+1$ or $i-1$ results in an extrapolant C' of C and $\overline{M_i}$. The algorithm QT

i	C	$\overline{M}$	C'
1	`true`	$[true,\ true,\ F_3]$	`true`
2	`true`	$[true,\ true,\ F_3]$	`true`
3	`true`	$[true,\ true,\ F_3]$	$z \geq 0 \wedge x \geq 0 \wedge y \geq 0$
2	`true`	$[true,\ \neg(x \geq 0 \wedge y \geq 0),\ F_3]$	$\neg(y \geq 0)$
3	$\neg(y \geq 0)$	$[true,\ \neg(x \geq 0 \wedge y \geq 0),\ F_3]$	$z \geq 0 \wedge -y - z + 1 \geq 0$
2	`true`	$[true,\ x < 0 \wedge y > 1,\ F_3]$	$x < 0 \wedge y > 1$
3	$x < 0 \wedge y > 1$	$[true,\ x < 0 \wedge y > 1,\ F_3]$	`false`
2	`true`	$[true,\ x < 0 \wedge y > 1,\ F_3]$	$x < 0 \wedge y > 1$
1	`true`	$[true,\ x < 0 \wedge y > 1,\ F_3]$	$x < 0$
1	`true`	$[x \geq 0,\ x < 0 \wedge y > 1,\ F_3]$	$x \geq 0$
2	$x \geq 0$	$[x \geq 0,\ x < 0 \wedge y > 1,\ F_3]$	`false`

Table 3.2: An example trace of the algorithm QT

strengthens M_i in each call and the last C being returned gives an equi-satisfiable quantifier-free formula of F_n.

We experimented with different instantiations for Linear Integer Arithmetic and reported experimental results of the variants. The detailed configurations are shown in Table 3.3.

Projection	**Extrapolation**
(A) Full quantifier elimination	(0) Trivial extrapolant
(B) Partial quantifier elimination	(1) NNF strengthening
	(2) SMT-TEST

Table 3.3: Different configurations for experimentation

Two distinct benchmark sets are generated with controlled sizes for experiments. These benchmarks come from marking formulas markT_{ij} in the Duration Calculus model checking algorithm so they are of deep nesting and challenging to solve. Benchmark set 1 has relatively small sizes; it is used to compare between

Benchmarks	**Number of formulas**	**Number of quantifiers**
Set 1	32	56-94
Set 2	64	69-768

Table 3.4: Generated benchmarks coming from the Duration Calculus model checking algorithm

different variants and Z3's current procedure. The experiment shows that multiple combinations e.g. A0 (full projection + trivial extrapolant) and A2 (full projection + SMT-TEST) are an order of magnitude faster than Z3. Benchmark set 2 consists of a number of big formulas for evaluating most competitive combinations above. Empirical evidence shows that a combination of full projection and SMT-TEST extrapolation (A2) scales well on these benchmarks from Duration Calculus model checking. While it is difficult to find a superior combination in all cases, the parameterization nature of the approach gives users more flexibility to choose suitable procedures depending on use cases.

The most efficient variants are exposed in a general purpose tactic in Z3. The algorithm is readily available to solve any problem that requires quantifier alternation. For our model-checking use case, this approach is complete, scalable and complementary to the heuristic-based solution in Section 3.1.

3.4 EHRA: Specification and Analysis of Energy-Harvesting Wireless Sensor Networks

Discrete-event simulation has been used to efficiently model energy consumption of WSN. Simulation can be done via domain-specific programming languages [61] or comprehensive simulation platforms [72]. Another complementary approach is to apply formal methods to modelling and analysis of WSN. Model checking tools have been employed to gain insights about WSN systems and protocols [16, 52]. However, the model checking approach suffers from scalability issues mainly due to state explosion. One may have to resort to simulation for large-scale networks.

Paper D presents a hybrid approach that utilizes both formal methods and simulation techniques for analyzing WSN. Different from most related work, we take into account energy harvesting, which adds an extra layer of complexity to the analysis.

An example network showing the challenge with energy-harvesting-aware protocols is given in Fig. 3.6, where we have a grid 7×7 of identical nodes. There is no node in a hole of size 3×3. Each node communicates with at most four neighbors which are its closest nodes vertically or horizontally.

A shade on a node, if present, has big influences on the ability to harvest energy. The node at position $(1, 1)$ is the base station and other nodes try to forward

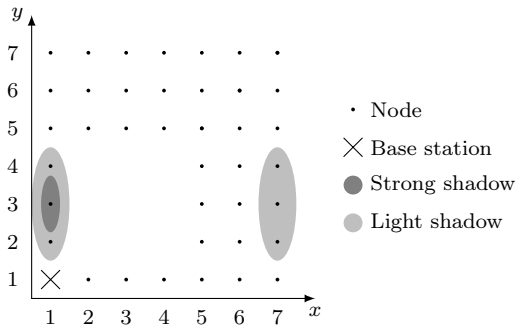


Figure 3.6: A network structure with shadowed nodes

observations to this base station. Messages are being passed through two routes along the axes in order to reach the base station. If the routes are not dynamically adapted based on energy changes, the network will be fragmented soon when critical nodes on 'bottleneck routes' get drained.

A generic modelling framework

We investigate a generic modelling framework for energy-harvesting-aware WSN. The goal is to examine a broad class of energy-harvesting-aware routing protocols through comprehensive models and detailed simulations. We are interested in global behaviors of the network during analysis e.g. how long does it take for the network to be fragmented, what are the routing trends of certain nodes and will they recover after getting drained?

Fig. 3.7 describes a typical sensor node with energy harvesting capabilities. Each node is represented by its identity and current physical state. The node can interact with the environment and record an observation at certain time. It also communicates with its neighbors by sending and receiving messages.

- $id, t \vdash_{\mathrm{O}} d, o$, where $t, d \in \mathbb{R}_{\geq 0}$: Node id can make the observation o at time $t + d$ where d is an amount of delay.

- $id, t \vdash_{\mathrm{H}} d, ps$, where $t, d \in \mathbb{R}_{\geq 0}$: Node id has physical state ps at time $t + d$.

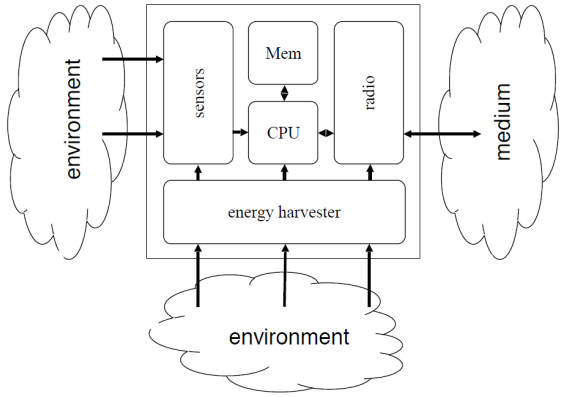


Figure 3.7: Architecture of an energy-harvesting sensor node

- $\bar{s}, id \vdash_{\mathrm{M}} id'$: Node id' can receive a message from node id in the network state $\bar{s}$.

Let S_{id} denote a node under consideration, where id is a unique identifier to recognize the node inside the network. Let PhysState, Observation and Msg be the sets of physical states, observations and messages respectively. The operations above can be modelled by the following relations:

- Sample the physical state: $\mathsf{samplePS}_{id} : S_{id} \times \mathsf{PhysState} \to S_{id}$
- Sense observation: $\mathsf{senseObs}_{id} : S_{id} \times \mathsf{Observation} \to S_{id}$
- Receive message: $\mathsf{treatMsg}_{id} : S_{id} \times \mathsf{Msg} \to S_{id}$

We model generic nodes as composite types consisting of basic operations and various energy and cost functions (see Fig. 3.8). These nodes are affected by network topology and a generic environment comprising of energy harvesting, rate of observation and communication medium, etc. Network operation is instantiated by plugging in a specific energy-harvesting-aware routing protocol.

Semantics of the networks is modelled as labelled transition systems. In Fig. 3.8, we show fundamental operations of a sensor node. A transition of a node is obtained by composition of its basic operations. Each routing protocol is described in many-sorted algebras so that its behaviors are clear and concise.

$id \in \mathsf{Id}$	unique node identifier
$o \in \mathsf{Observation}$	observation
$ps \in \mathsf{PhysState}$	physical state
$cs \in \mathsf{CompState}$	computational state
$as \in \mathsf{AbsState}$	abstract state
$m \in \mathsf{Msg}$	message having one of the forms:
$\mathsf{obsMsg}(dst, o)$	observation message with node $dst \in \mathsf{Id}$ as destination
$\mathsf{nbMsg}(src, as)$	neighbor message from node $src \in \mathsf{Id}$

$$
\begin{array}{lcl}
\mathsf{next} & : & \mathsf{CompState} \to \mathsf{Id} \\
\mathsf{updateEnergyState} & : & \mathsf{CompState} \times \mathsf{PhysState} \to \mathsf{CompState} \\
\mathsf{updateRoutingState} & : & \mathsf{CompState} \to \mathsf{CompState} \\
\mathsf{consistent?} & : & \mathsf{CompState} \to \{\mathrm{true}, \mathrm{false}\} \\
\mathsf{abstractView} & : & \mathsf{CompState} \to \mathsf{AbsState} \\
\mathsf{updateNeighbourView} & : & \mathsf{CompState} \times \mathsf{Id} \times \mathsf{AbsState} \to \mathsf{CompState} \\
\mathsf{transmitChange?} & : & \mathsf{CompState} \times \mathsf{CompState} \to \{\mathrm{true}, \mathrm{false}\}
\end{array}
$$

Figure 3.8: Fundamental types and operations

The formal semantics forms the basis of modelling WSN in a precise manner. In the meantime, the modelling formalism is used as a conceptual framework to study energy-harvesting-aware protocols. The analysis phase is performed through discrete-event simulation.

The protocols

In Paper D, we have experimented with three distinctive protocols including Directed Diffusion (DD) [39], Energy-Aware Routing (EAR) [65] and Distributed Energy-Harvesting-Aware Routing (DEHAR) [40]. While DD chooses a route by shortest path (number of hops) to the base station, EAR finds a neighbor with a shorter distance probabilistically by their energy levels. Moreover, DEHAR uses energy levels and faithful adjustments in order to propagate information about energy levels. One can say that these protocols are ordered in the progression of energy awareness, where DEHAR is the only protocol allowing detours in the routing of observations.

The protocols are examined on the same network under the same environment. We simulate the network on 30 days of simulation time. The environment has 12 hours of full light and 12 hours of no light every day. A node in light shadow can

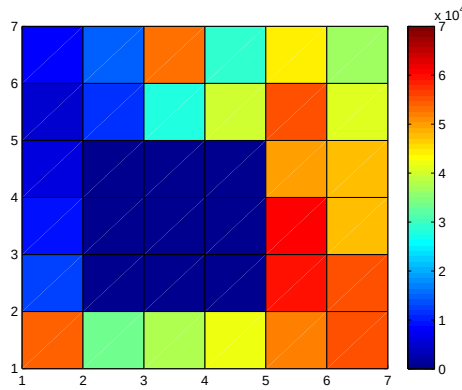


Figure 3.9: Amount of messages of DEHAR nodes after 720h

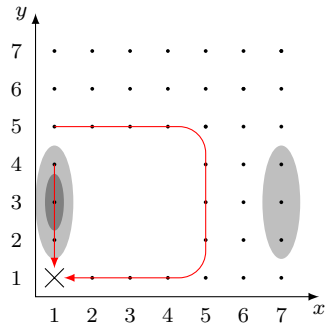


Figure 3.10: Routing trends of DEHAR nodes after 720h

harvest up to 75% amount of solar energy and another in dark shadow is only able to at most harvest 25% of solar energy. We assume to sample energy once in every 1800 seconds and observe the environment once in every 900 seconds. The experimental results show that DEHAR is able to harvest 50% more energy than DD and 40% more than EAR. Moreover, DD had 1.5× as many drained nodes as EAR had and 3.5× as many as DEHAR had. The experiments also indicate that DEHAR nodes are able to recover from drainage and messages are detoured along long paths while network fragmentation occurs (see Fig. 3.10).

The work lays the foundation for more comprehensive analysis based on proposed formal semantics for WSN. As future work, we may investigate other protocols

as case studies and attempt formal verification via statistical model checking.

3.5 νZ - An optimizing SMT solver

Earlier in this chapter, we have considered the use of SMT solvers for solving satisfiability problems. Many problems can be encoded as a SAT problem involving different fragments of first-order logics. By using SMT solvers, we are able to focus on the problem at hand and exploit the power of modern SMT solvers. However, not all problems fit into a satisfiability framework in a natural manner. Optimal solutions are often desired for many problems of Cyber-Physical nature.

Many users have built custom loops around SMT solvers to achieve optimal assignments for objective functions. But it is more efficient to handle optimization in a native way "inside" SMT solvers. SMT community has done a lot of work on MaxSAT [14, 46, 50]. By MaxSAT, we mean the optimization problem where the number of satisfiable formulas is maximized for a given set of Boolean formulas. The *optionally satisfiable* constraints are called "soft constraints" to contrast with the hard constraints where this flexibility is not allowed. Furthermore, some progress has been made for arithmetic optimization in SMT solvers [62, 13, 44].

This section introduces νZ, an extension to Z3, that provides a portfolio of approaches to solve arithmetic optimization, MaxSAT and combinations thereof. The tool supports multiple ways of combining objectives e.g. independently, lexicographically or by using Pareto fronts.

A motivating example

We use a transportation problem in the context of Dynamics AX, an enterprise resource planning software product from Microsoft, in order to motivate the development of νZ. One of the core problems in Dynamics AX is to distribute all shipments into trucks under some specific constraints. For example, trucks have limited weights and volumes; certain goods cannot be packed together due to safety requirements. The objectives are to minimize the number of trucks used and reduce transportation costs as much as possible.

The scenario is an instance of the classic bin packing problem. There is a lot of

Shipments	Weight(kg)	Volume(m^3)	Requirement
1	400	300	Dry
2	300	350	Fresh
3	220	160	Dry

Table 3.5: Shipments that requires transportation

Trucks	Weight(kg)	Volume(m^3)	Requirement	Initial cost	Extra cost
1	777	700	Fresh	\$100	\$20
2	450	1000	Dry	\$120	\$30
3	600	460	Dry	\$130	\$10

Table 3.6: Available trucks for transportation

research on solving bin packing variants e.g. based on constraint programming techniques [68] or based on specialized algorithms [41]. In this particular example, we focus on flexibility i.e. how easily constraints can be revised when changing requirements.

Fig. 3.11 describes the bin packing constraints that form the satisfiability problem. The expressiveness of SMT solving is really helpful here. Most constraints can be formulated as Boolean combinations of linear constraints over 0-1 variables. Let 0-1 variable x_{ij} denote that shipment i is packed into truck j and 0-1 variable y_j denote that truck j is in use. The first block of constraints shows that only one in three 0-1 variables x_{i1}, x_{i2}, x_{i3} is exclusively equal to 1. The second block of constraints means a truck y_j is in use if a shipment is packed into it i.e. y_j takes the largest value in those of x_{1j}, x_{2j}, x_{3j}. The last block consists of weight capacity constraints. They denote that total weights of shipments cannot exceed weights of corresponding trucks. There are similar constraints on volume capacity, but they are not shown here.

Furthermore, optimization goals can be expressed as lexicographical combinations of multiple objective functions. In Fig. 3.12, we formulate two objectives: one based on a group of soft constraints called *unused_ trucks* and the other based on a linear sum of 0-1 variables. While the first objective is to maximize number of unused trucks, the second one denotes minimization of total transportation cost. For simplicity, total transportation cost is the sum of initial costs of deploying trucks and extra costs of handling individual shipments. These concrete costs are available in Table 3.6. These objectives show the flexibility in expressing optimization problems, one of key targets in developing νZ.

```
(assert (= (+ x_11 x_12 x_13) 1))
(assert (= (+ x_21 x_22 x_23) 1))
(assert (= (+ x_31 x_32 x_33) 1))

(define-fun imax ((a Int) (b Int)) Int (if (> a b) a b))
(assert (= y_1 (imax (imax x_11 x_21) x_31)))
(assert (= y_2 (imax (imax x_12 x_22) x_32)))
(assert (= y_3 (imax (imax x_13 x_23) x_33)))

(assert (<= (+ (* 400 x_11) (* 300 x_21) (* 220 x_31)) (* 777 y_1)))
(assert (<= (+ (* 400 x_12) (* 300 x_22) (* 220 x_32)) (* 450 y_2)))
(assert (<= (+ (* 400 x_13) (* 300 x_23) (* 220 x_33)) (* 600 y_3)))
```

Figure 3.11: The constraints of the transportation planning example

```
(assert-soft (= y_1 0) :id unused_trucks)
(assert-soft (= y_2 0) :id unused_trucks)
(assert-soft (= y_3 0) :id unused_trucks)

(minimize (+ (* 100 y_1) (* 20 (+ x_11 x_21 x_31))
             (* 120 y_2) (* 30 (+ x_12 x_22 x_32))
             (* 130 y_3) (* 10 (+ x_13 x_23 x_33))))

(set-option :opt.priority lex)
(check-sat)
```

Figure 3.12: Two optimization goals of the transportation planning example

Inspired by the example above, νZ adds a few new commands to SMT-LIB standard:

- `(maximize t)` or `(minimize t)` - instruct the solver to maximize or minimize `t` where the term `t` can be either Integer, Real or Bit-vector.
- `(assert-soft` F `[:weight n | :dweight d] [:id id])` - assert soft constraint F, optionally with an integral weight `n` or a decimal weight `d`. Soft constraints can be furthermore tagged with an optional name *id*. This enables combining multiple different soft objectives.

For combining objectives, νZ provides three different ways:

- **Boxes**: each objective is optimized independently subject to the original constraints.
- **Lexicographic combinations**: for two maximization objectives t_1 and t_2, it finds a model M where $(M(t_1),\ M(t_2))$ are maximized lexicographically.
- **Pareto fronts**: given two maximization objectives t_1 and t_2, the set of Pareto front is a set of models M_i where none of the models dominates the others.

For this particular example, an optimal solution is to use truck 1 to carry shipment 1 and 3, and truck 2 to carry the remaining shipment. It means that we have truck 3 to spare and settle down with a minimum transportation cost of $290.

We now consider the νZ architecture and main ideas behind the implementations. In addition to MaxSMT, νZ provides a portfolio of multiple approaches for solving arithmetic optimization problems and their combinations. In Fig. 3.13, νZ components are described in more detail.

- First, constraints are translated to weighted sums of 0-1 variables i.e. Pseudo-Boolean variables. With the use of objectives, we have an optimization problem on Pseudo-Boolean domain i.e. Pseudo Boolean Optimization (PBO).
- Multiple objective functions are analyzed to identify a relevant orchestrator in charge of computing optimal results.

- For a pure arithmetic instance, we use **OptSMT** module, which provides complementary approaches for solving arithmetic optimization.
- If the instance involves in soft constraints, **MaxSMT** module is in charge of finding a suitable algorithm for solving the MaxSMT problem in an efficient way.
- Depending on inputs' characteristics, **OptSMT** and **MaxSMT** choose to use Z3's **SMT solver** or **SAT solver** under the hood.
- Pseudo Boolean and Cost solvers can be implemented efficiently as satellite modules of **SMT solver** without too much complication.

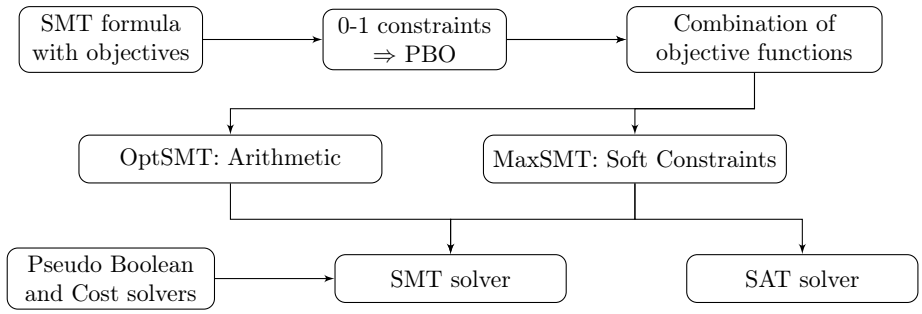


Figure 3.13: νZ system architecture

OptSMT: Arithmetic Optimization

Algorithm 4 shows a basic optimization algorithm for arithmetic fragments such as Linear Integer Arithmetic, Linear Real Arithmetic and Difference Logic (DL). If we have F and t to be an input formula and an objective to maximize respectively, the algorithm works as follows.

- Choose L as a set of literals (from F) that imply F and construct a partial model that satisfies L.
- Maximize t as much as possible given the partial model.
- Remember the current best objective and re-iterate until there is no more distinctive set of literals.

In order to maximize an objective locally, linear real/integer arithmetic optimization module uses primal Simplex to compute optimal values. For difference logic, an important catch is that optimization goals may not be in difference logic. Therefore, we do not block objectives directly but assert negations of literal values to ensure convergence. Moreover, νZ uses non-standard numbers to conveniently represent unbounded objectives. Each number takes a value of the form $c\infty + b + \epsilon a$ where a, b, and c are rational numbers and ∞ is larger than any finite constant; this representation also supports all Simplex operations.

Input: Formula F and Objective t to maximize
Output: Maximal value v, such that $v = t \wedge F$ is satisfiable
$v \leftarrow -\infty$
while F *is satisfiable* **do**
  Let L be a set of literals (from F) that imply F.
  if t *is unbounded in* L **then**
    return ∞
  end
  Let M be an interpretation that satisfies L and maximizes t
  $v \leftarrow \max(v, M(t))$
  $F \leftarrow F \wedge t > v \wedge \neg \bigwedge L$
end
return v

Algorithm 4: Sequential Bound Increase

MaxSMT: Soft Constraints

The weighted MaxSMT problem is defined as follows. Given a formula F_0, a set of formulas $F_1, \cdots, F_n$ and corresponding numerical weights $w_1, \cdots, w_n$, find a subset $I \subseteq \{1, \cdots, n\}$ such that:

- $F_0 \wedge \bigwedge_{i \in I} F_i$ is satisfiable.
- The reward $\Sigma_{i \in I} w_i$ is maximized.

The tool νZ implements a portfolio of multiple engines for MaxSMT. These include WMax [50], MaxRes [49], BCD2 [46] and MaxHS [18] algorithm. We present here one of the simplest algorithms, namely WMax. For more details on other engines, we refer to Paper E and Paper F.

WMax asserts F_0 and $F_i \vee p_i$ where p_i is a new propositional variable. The idea of p_i is to track satisfiability of the soft constraint F_i. We maintain a current cost c and update it whenever p_i is switched to `true`. If c exceeds the minimum cost c^*, the corresponding partial model is blocked so that its search space should no longer be explored. When all p_i are assigned and $c < c^*$, we can update the new minimum cost and block the corresponding assignment. WMax is a very simple algorithm that can be implemented outside Z3's SMT core. The advantage of WMax is that it can be interrupted at any point with a current upper bound. It also means that we can return current best values after a given timeout, which is crucial for many real-world scenarios.

Experimental results

In Fig. 3.14, we compare νZ with the three best solvers in each category of MaxSAT competition 2014. Since νZ did not enter the competition, we replicate results for νZ using given benchmarks on similar test environments. Presently νZ is not fine-tuned as the other solvers are, but it performs quite respectably. The graphs show the number of instances each solver is able to solve under 1800-second timeout. The tool νZ can solve up to 400 instances for Partial MaxSAT and 350 instances for Weighted Partial MaxSAT under the given timeout. These results are much better than those of many solvers that participated in the competition.

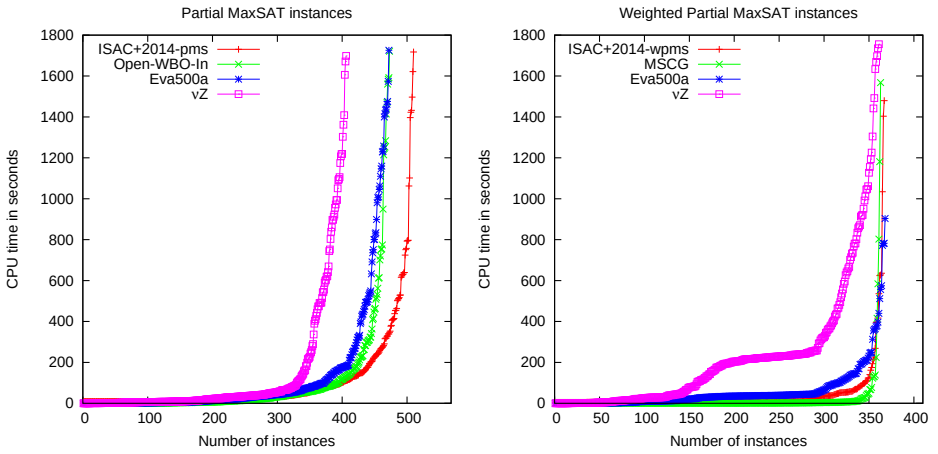


Figure 3.14: MaxSAT 2014 competition evaluation results

We performed selected evaluation of νZ on different problems from Z3 users. The

Source	Category	Solved instances	Time
Longest Paths (easy)	MaxSAT	8/8	<0.05s
Longest Paths (medium)	MaxSAT	34/34	1-36s
DAL Allocation challenge	PBO	96/96	0.02-6s
Symba	LRA	2435/2435	0.2s-36s
OptiMathSAT	LRA	9/9	0.5-20s

Table 3.7: Evaluation of νZ on selected examples

Longest Paths benchmarks come from one of our collaborators. DAL Allocation challenge is a recent competition for PBO solvers. We use LRA benchmarks from Symba [44] and OptiMathSAT [62], the two solvers that currently support linear arithmetic optimization. The selected evaluation shows encouraging results across multiple categories including MaxSAT, PBO and LRA.

We summarize here the main contributions:

- Extend SMT-LIB language with relevant extensions for modelling optimization problems in a powerful and flexible way.
- Support portfolios of algorithms for MaxSMT, arithmetic optimization and their combinations, that are readily available out of the box.
- Provide a foundation for modelling different Cyber-Physical problems that lend themselves to optimization.

Chapter 4

Two cases

In this chapter, two case studies in the domain of Cyber-Physical Systems are presented. One goal is to examine different flavors of CPS in the real world. Another goal is to evaluate our modelling and analysis techniques on various settings. For each case, we present a high-level description and a detailed analysis, then briefly discuss experimental results.

4.1 Smart Meter

With Cyber-Physical Systems becoming more pervasive, *smart home appliances* become more and more popular. The main idea is that various home appliances can be well-connected and self-regulated without human intervention. We present here a concrete use case of smart home appliances, called Smart Meter. The use case arose from a working group on CPS with our colleagues, Flemming Nielson and Nataliya Skrypnyuk, at DTU Compute.

The Smart Meter interface consists of two main components, a washing machine and a dishwasher. Each component is controlled by a corresponding task. Before

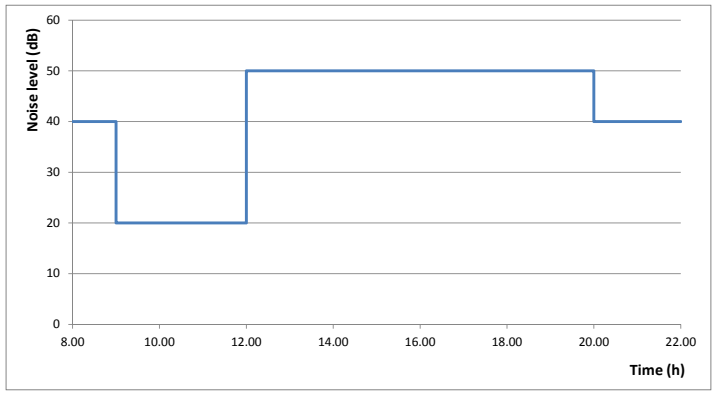


Figure 4.1: Users' maximal tolerance w.r.t. noise levels

leaving home for work, the home owner starts the Smart Meter interface and wishes to complete the two tasks at the end of the day. The Smart Meter controller has to deduce a schedule for the tasks where maximal level of noise tolerance and maximal power rate are not exceeded. The objective is to minimize the price the home owner has to pay in the end.

Regarding the flavor of CPS, the setting here is quite simple. The goal is to determine *coordination* between tasks in order to give informed decisions for scheduling smart-home appliances. The control operation is offline in the sense that the duration is fixed and the appliances are operated by specified plans. For future work, online control is an interesting direction where there is much more *interaction* between components and schedules have to be adjusted based on changes.

For this concrete use case, the maximal power rate is 2.2kW and users' maximal tolerance w.r.t. noise level is specified in Fig. 4.1. The washing machine runs for 2 hours with a power rate of 2kW and a noise level of 41dB. The dishwasher operates for 2 hours, requires 1kW supply and emits 39dB w.r.t. noise levels. Energy supply has a given pricing structure (see Fig. 4.2). The time period for scheduling is 14 hours between 8:00 and 22:00 on the given day.

First, the problem is modelled by a discrete version of DC. This allows for a focus on the essence of the problem when DC acts as a domain specific language to

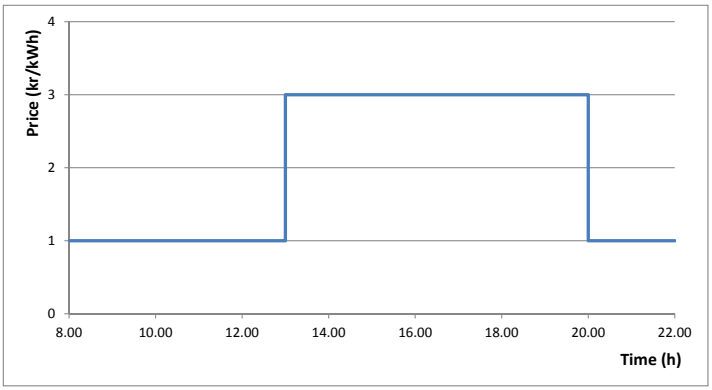


Figure 4.2: Pricing structure of energy supply

express the problem at hand. Second, DC formulas are encoded to Linear Integer Arithmetic using automatic transformations. This step helps us leverage the powerful SMT solving engines to efficiently solve the formulas. The methodology is similar to what has been used throughout Paper A.

Formalization in Duration Calculus

Let W and D denote state variables for the washing machine and the dish washer respectively

$$\mathrm{W},\ \mathrm{D} : Time \to \{0,\ 1\}$$

where $\mathrm{W}(t) = 1$ when the washing machine is used at time t. A similar explanation applies for D.

Energy costs ($\mathrm{E_C}$) and noise levels ($\mathrm{L_{Noise}}$) are discrete functions of time

$$\begin{aligned} \mathrm{E_C} &: Time \to \{1,\ 2,\ \cdots,\ N\} \\ \mathrm{L_{Noise}} &: Time \to \{1,\ 2,\ \cdots,\ M\} \end{aligned}$$

where $\mathrm{E_C}(t) = n$ and $\mathrm{L_{Noise}}(t) = m$ when the energy cost and noise level are at

the given amounts at time t. Here N and M are maximal values of energy costs and noise levels respectively.

Prices can be expressed as linear sums

$$P \stackrel{\text{df}}{=} \Sigma_{k=1}^{N} d_k \int(\mathrm{D} \wedge (\mathrm{E_C} = k)) + \Sigma_{k=1}^{N} w_k \int(\mathrm{W} \wedge \mathrm{E_C} = k))$$

where d_k and w_k are constants. The pricing structure for energy consumption is provided in Fig. 4.2. The state expression $D \wedge (\mathrm{E_C} = k)$ is true (i.e. equals to 1) at time t if the dishwasher is running at time t and the energy cost at time t is k. The state expression $W \wedge (\mathrm{E_C} = k)$ has a similar explanation.

The corresponding tasks can be formulated as follows

$$\begin{aligned} TS \stackrel{\text{df}}{=} & (\llbracket \neg \mathrm{D} \rrbracket^* \frown (\llbracket \mathrm{D} \rrbracket \wedge l = 2h)) \frown (\llbracket \neg \mathrm{D} \rrbracket^*) \wedge \\ & (\llbracket \neg \mathrm{W} \rrbracket^* \frown (\llbracket \mathrm{W} \rrbracket \wedge l = 2h)) \frown (\llbracket \neg \mathrm{W} \rrbracket^*) \end{aligned}$$

where each task has to be scheduled for 2 hours without interruption.

We can model limits on power supplies and on noise levels.

$$\begin{aligned} PS &\stackrel{\text{df}}{=} \Box(l = 1 \Rightarrow 2 \int \mathrm{W} + 1 \int \mathrm{D} \leq 2.2kW) \\ NL &\stackrel{\text{df}}{=} \Box(l = 1 \Rightarrow 41 \int \mathrm{W} \oplus 39 \int \mathrm{D} \leq \mathrm{L_{Noise}}) \end{aligned}$$

The operation $\oplus$ is special addition on dB.

Prices and noise levels can be modelled as follows.

$$\begin{aligned} PSNL \stackrel{\text{df}}{=} & ((l = 5 \wedge \mathrm{E_C} = 1) \frown (l = 7 \wedge \mathrm{E_C} = 3) \frown (l = 2 \wedge \mathrm{E_C} = 1)) \\ & \wedge ((l = 1.5 \wedge \mathrm{L_{Noise}} = 40) \frown (l = 2.5 \wedge \mathrm{L_{Noise}} = 20) \\ & \frown (l = 8 \wedge \mathrm{L_{Noise}} = 50) \frown (l = 2 \wedge \mathrm{L_{Noise}} = 40)) \end{aligned}$$

We have a bounded model checking problem with a duration of 14 hours and the main formula is $\phi \stackrel{\text{df}}{=} TS \wedge PS \wedge NL \wedge PSNL$. The objective is to schedule the tasks and minimize price P.

Translation to SMT solving

Different approaches have been taken for the problem of scheduling smart home appliances. Sou et al. used a detailed mathematical model to solve the problem using Mixed Integer Programming (MIP) [70]. The approach is complex due to the rigidity of MIP that only allows conjunctions of linear constraints. On a related note, Sheini et al. proposed a SAT-based approach to represent temporal constraints [69]. This approach has more flexibility in terms of describing complex relationships thanks to the use of expressive logical formulas.

The idea of applying BMC in the context of DC is not new. An elegant BMC encoding for a pure Boolean fragment of DC was introduced in [27]. Another approach for BMC of a discrete DC fragment was proposed in [74]. The paper described a reduction to timed automata and showed efficiency of the approach using Uppaal model checker. In this section, we investigate bounded model checking in another discrete variant of DC allowing linear sum, which is slightly more expressive than that of [27].

Given a natural number K and formula ϕ, one can find a model of size K for ϕ if such a model exists. The idea is to reduce the problem to a SAT problem for Linear Integer Arithmetic. For each state variable S, we introduce K variables $S_k \in \{0,\ 1\}$ denoting the value $S(k)$ at time $k \in \{0,\ 1,\ \cdots,\ K-1\}$. For each subformula ψ of ϕ and each interval $[i,\ j]$ where $0 \leq i \leq j < K$, a variable $F_{\psi,\ i,\ j} \in \{0,\ 1\}$ is introduced to denote the truth value of ψ in the given interval. We can translate in the polynomial time the encoding of ϕ and K to a quantifier-free formula $lin_{\phi,\ K}$ in linear arithmetic having P_k and $F_{\psi,\ i,\ j}$ as Boolean variables.

Experimental results

We implemented two versions of the encoding. An initial version was done using plain Z3 SMT solver and a custom optimization loop atop. This version was also used as a motivation for creating νZ. Another variant was built on top of νZ; the encoding was posed as a Pseudo-Boolean Optimization problem that is well-supported by νZ.

For this particular case, we have $d_k = 2$ and $w_k = 1$ where $1 \leq k \leq 3$ for the above-mentioned constants. An optimal assignment is to run the dishwasher between 10:00 and 12:00 and schedule the washing machine between 18:00 and

20:00 on the given day. The minimal price is 10 kr under the given constants.

Both two implementations completed their runs in a fraction of a second. The native νZ variant is twice as fast as the other implementation. The gap is not too big since the size of the problem is small. We implicitly use a granularity of half an hour for both variants. Consequently, there are not many steps to discretize for a span of 14 hours. We have not tried out with problems of bigger sizes; we suspect that the benefit of using native optimization procedures might be more significant for those cases. Moreover, the native νZ version is also much more concise and easier to derive thanks to excellent support for optimization.

Summary

We summarize the lessons learned from this section.

- The example demonstrates the adequacy of DC as a domain specific language. DC allows users to focus on modelling aspects and to delegate tedious transformations to underlying logics.
- It also validates the feasibility of the approach to use DC for modelling and LIA for analysis.
- The case study also shows the strength of νZ in expressing optimization problems and solving those in an efficient manner.
- We have had a single validation with a simple setup. A next step would be to consider a more extensive case study where granularity could be controlled.

4.2 Formal Design Space Exploration for Wireless Sensor Networks

This case study uses a similar setup for WSN as in other parts of the thesis. Each node has a limited memory capacity and it maintains a residual energy. Residual energy decreases when the node consumes energy during sending, receiving or making observations. Residual energy increases when the node harvests certain amount of energy from the environment. We assume that each observation has equal size one in terms of memory usage. Memory usage grows when the

node makes observations or receives messages. Another assumption is that a sensor node sends the whole amount of observations in the memory to one of its neighbors.

The use case comprises both *interaction* and *coordination* aspects. The setting is complex. Each node records observations by interacting with the environment, and the node needs to communicate with its neighbors and adjusts its behaviors accordingly. The *interaction* and *coordination* aspects are being influenced by harvested energy coming from the environment, making the problem even more challenging for rigorous analysis.

We consider the following problem. Given an observation frequency, a period length should be determined so that there is no energy depletion and the network operates optimally according to certain metrics. A special problem is to dimension each node so that it functions properly and has less risks of being drained under specified parameters.

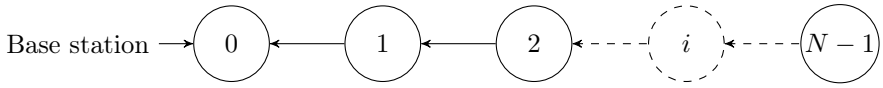


Figure 4.3: A WSN consisting of nodes in a chain topology

For a given network topology, a key question is:

"How long does the network operate until a node's energy is depleted?"

We also would like to investigate the relationship between amount of observations and distances to the base station. For instance, what is the maximal number of nodes one can put to the topology so that there is no node depletion? These are interesting *design space exploration* questions in order to gain more insights about WSN through quantitative analysis. Answers to these questions may be incomprehensible for complex topologies. Therefore, we use a specific single chain topology (see Fig. 4.3) in order to achieve a deeper understanding of the problem. Hopefully the lessons learned can be generalized for other topologies.

We would like to avoid memory overflow in sensor nodes. The whole network is examined until a node dies. On the chain topology, the implication is that any node further away from the base station becomes unreachable. The nodes are numbered so that node 0 is the base station where all observations should be gathered, and node i is a neighbor of node $i + 1$ and node $i - 1$.

The questions above constitute an optimization problem that fits nicely with the paradigm νZ pursues. This opens up a big opportunity to use the optimization framework presented in Paper E and Paper F for quantitative analysis.

Formal analysis

In Paper D, we have presented a simulation-based approach for analyzing global behaviors of WSN. While the approach is efficient, it cannot provide any absolute guarantee on analysis results. In this setting, we are interested in a more formal analysis. We have attempted to use timed automata model checking for formulating WSN, but the approach was not able to scale further than a few nodes.

This section still focuses on formal analysis of global properties of WSN. The main idea is to use *counting semantics* to support analysis at a high-level abstraction. Global analysis gives insights about WSN behaviors; it results in some kinds of templates in terms of node behaviors. At the abstract level, frequencies rather than sequences of observations and messages are considered. At the node level, one can do local analysis to synthesize concrete node behaviors.

Under a counting semantics, we do not look into behaviors of a node at a specific time instance but have a general view on node behaviors during concrete time periods. For this use case, a fixed number of periods are employed to determine a period length that satisfies the end goals. We can consider this as a system-level encoding where non-optimal schedules of the network are ruled out. For instance, a concrete period length is determined where we can maximize the number of observations.

We devise a **two-step optimization** scheme, where the first phase solves an abstract problem and the second phase deals with a concrete version. The abstract semantics acts as a design space exploration step to rule out infeasible combinations of parameters. The concrete semantics finds an exact model based on values of the counting variables.

Let us consider a chain of N sensor nodes and examine the network in fixed K periods of length T. Let i be an index of a node ($0 \leq i \leq N-1$) and j be an index of a period of interest ($0 \leq j \leq K-1$).

We make use of counting variables for each node in these periods. Instead

of determining an exact time instance where an action occurs, the number of occurrences of those actions are considered for a given period. The purpose is to abstract away from overwhelming details and to understand the network at a high-level abstraction. Recalling from Chapter 1, automata-based and logic-based approaches often need to rule out irrelevant information from models to make formal analysis feasible. In this section, we use counting semantics to suppress irrelevant details, a technique which has been shown to be useful in Paper A.

The counting variables are:

- S_{ij} denotes the number of occurrences of messages sent at node i during j-th period.
- R_{ij} denotes the number of occurrences of messages received at node i during j-th period.
- O_{ij} denotes the number of occurrences of observations at node i during j-th period.

Constraints on these counting variables are formalized as follows.

- The amount of sends, receives and observations should not exceed the length of a given period.
$$\mathrm{S}_{ij} + \mathrm{R}_{ij} + \mathrm{O}_{ij} \leq T$$
Thus we assume that a node can execute at most one action at a time instance.
- The amount of messages sent at a node should be the same as the amount of messages received at its neighbor (no message loss).
$$R_{(i-1)j} = \mathrm{S}_{ij}$$
- The cumulative amount of messages sent at a node up to the current period should be less than or equal to the cumulative amount of messages received and observed at the node.
$$\sum_{j=0}^{k} \mathrm{S}_{ij} \leq \sum_{j=0}^{k} (\mathrm{O}_{ij} + \mathrm{R}_{ij}) \text{ where } 0 \leq k \leq K-1$$

- The cumulative amount of messages sent at a node should at least be equal to the cumulative amount of received messages and observations until the previous period.

$$\sum_{j=0}^{k} \mathrm{S}_{ij} \geq \sum_{j=0}^{k-1} (\mathrm{O}_{ij} + \mathrm{R}_{ij}) \text{ where } 1 \leq k \leq K-1$$

  This constraint assumes that an observation or message has to be sent within time T.

- There is at least one observation per period per node.

$$\mathrm{O}_{ij} \geq 1$$

  It ensures that there is some activity on a node in any period.

Experimental results

We experimented with a chain of 6 nodes. The nodes have the same energy-harvesting pattern i.e. half of a day without energy harvesting and the other half with 20% total energy capacity that is available to harvest in every hour. Residual and harvested energies were recorded. Amounts of message exchanges were also monitored.

In Fig. 4.4, one can see that residual energy of node 1 is close to be depleted at 15^{th} period. This node is the main bottleneck of the topology since it guards the only route to the base station. Indeed node 1 runs out of energy if the chain is analyzed for 17 periods. Residual energy of a node increases proportionally with its distance to the base station.

The synthesized period length is 205 seconds. Fig. 4.5 shows the amount of messages sent, received and observed during the 15 periods. The figure also acts as a sanity check where one can inspect that the relationships between the variables S_{ij}, O_{ij} and R_{ij} are meaningful.

Note that the results only provide a general picture of nodes' behaviors; there is no information concerning on when an action actually happens. We have not investigated concrete scheduling of the network based on these generic configurations. In general, the generic configurations are good to show general trends on the chain topology i.e. messages are denser while being close at the

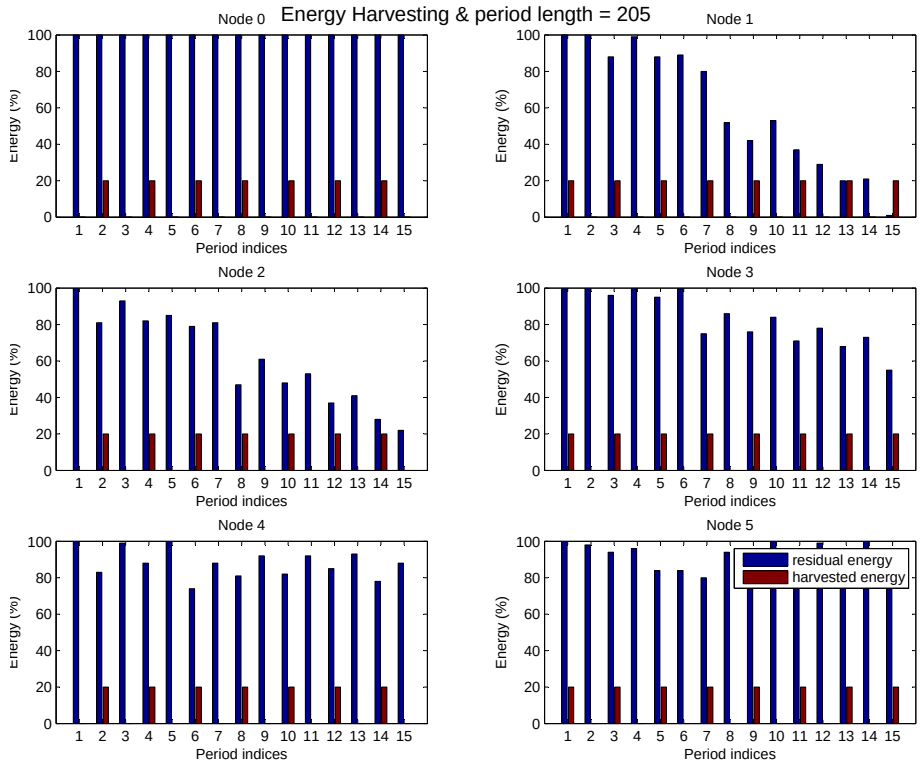


Figure 4.4: Residual and harvested energy during 15 periods

base station. We are able to highlight these trends without compromising the abstract encoding, which is designed to handle large networks.

Summary

There are a number of bodies of research on formal analysis of WSN. In [16], sensor nodes are modelled in HyTech where their safety properties are analyzed. For a large WSN, a network of Hybrid Automata is simulated where energy consumption is also taken into account. Real-Time Maude is used to analyze the Optimal Geographical Density Control algorithm for WSN in [52]. Model-

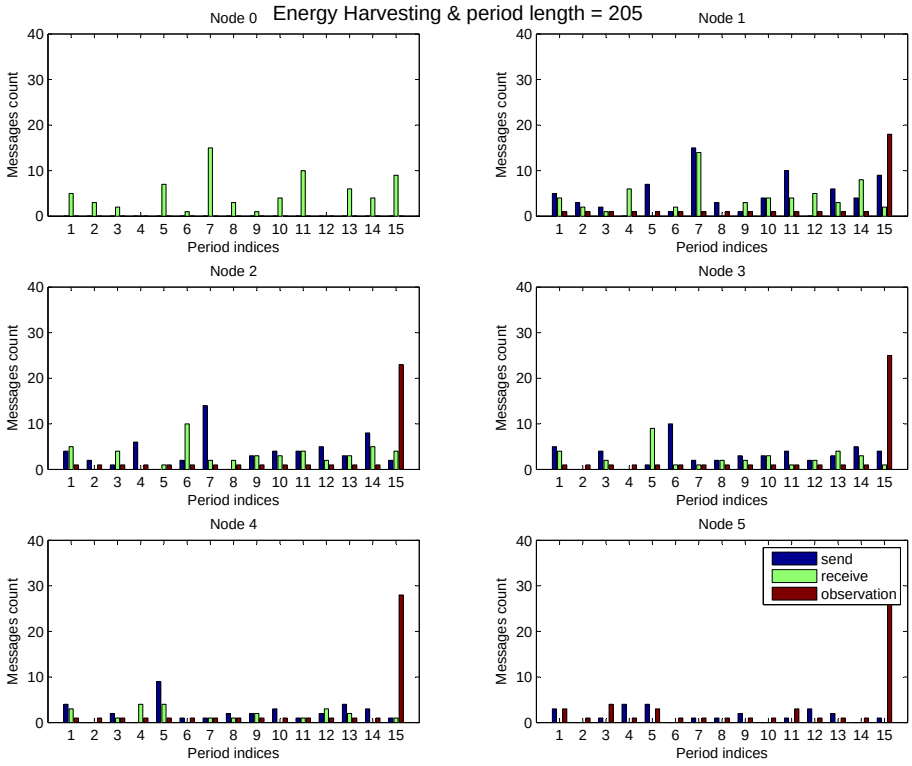


Figure 4.5: Number of messages sent, received and observed during 15 periods

checking techniques are used to verify functional properties and performance evaluation is performed via the corresponding simulator as well. Note that these related works do not consider energy harvesting.

In this section, we have shown that SMT techniques are useful for analyzing global behaviors of WSN in connection with energy harvesting. The case study has given us a number of useful lessons.

- The abstract encoding is helpful to understand global behaviors of the network.
- *Counting semantics* is a powerful technique to obtain a high-level but useful

abstraction.

- We are able to obtain answers for the design space exploration questions earlier.
- We would like to generalize the analysis to other topologies. The challenge is to model communication amongst nodes.

Chapter 5

Conclusions

Before giving a detailed summary of the work, we would like to recapitulate the goals set out in Section 1.3.

- Develop theories, tools and techniques for Duration Calculus, that are geared towards high-level modelling and analysis of Cyber-Physical Systems (CPS).
- Develop theories, tools and techniques for SMT solving, that are geared towards analysis of CPS.
- Develop modelling and analysis techniques for Wireless Sensor Networks with special focus on energy harvesting.

Summary

In previous chapters, we have presented our main contributions and relevant case studies. In Paper A, we started with the model checking problem on Duration Calculus where the high complexity prohibited practical tool support. *Counting*

semantics was employed to construct an efficient reduction of the problem to Linear Integer Arithmetic. Thanks to Guarded Normal Form, generated formulas can be solved efficiently using Z3 SMT solver. The *counting semantics* technique was used later in Wireless Sensor Network case study, resulting in a scalable abstract encoding.

Motivated by the model checking problem, we investigated different ways to solve the problem in an efficient manner. First, we examined multicore parallelism as a means to exploit extra computation powers for the decision procedures of LIA in Paper B. The experiments resulted in encouraging speedups for Cooper's algorithm and exact shadow of Omega Test, which have positive influences on efficiency of the model checking algorithm. More importantly, the parallelization process can be applied to other advanced tree processing algorithms. Second, algorithms for quantifier elimination in context of satisfiability were investigated in Paper C. We proposed a generic algorithm for efficiently solving quantified formulas with deep nesting of alternating quantifiers. The algorithm along with multiple instantiations were implemented as a common tactic in Z3 SMT solver. While the algorithm has wide applicability, it also brings further gains to directly improve the model-checking algorithm in Paper A.

We then shifted our focus to a concrete class of CPS i.e. Wireless Sensor Network (WSN). Paper D presented a modelling framework of energy-harvesting-aware WSN with formal semantics. Using this formal model, one is able to analyze different routing protocols in a consistent way using similar network topologies and underlying conditions. Three routing protocols were instantiated via many-sorted algebras, including Directed Diffusion (DD), Energy-Aware Routing (EAR) and Distributed Energy-Harvesting-Aware Routing (DEHAR). Then a discrete-event simulator was developed as a basis to compare these protocols. The goal was to examine global behaviors of WSN under a concrete protocol. The paper was based on a simulation-based approach, which allows a fine-grained way of modelling and analyzing systems. Experimental results have shown that DEHAR algorithm turned out to be superior. Its distinctive feature is the ability to adapt so that drained nodes could be recovered by routing messages longer paths.

After that, we revisited SMT solving and investigated their use in CPS modelling and analysis. Many problems of CPS nature lend themselves to optimization. Paper E and Paper F presented two aspects of an extension to Z3 for optimization i.e. νZ. While Paper E focuses on details of algorithms and lessons learned while developing νZ framework, Paper F highlights flexibility and powers of optimization in the context of SMT. In Chapter 4, we examined two case studies

which both use νZ as the main analysis engine. Experiments have shown promising results concerning the use of νZ in the analysis of CPS.

We have achieved our goals to a large extent. The Duration Calculus formalism has been instrumental to inspire us to investigate different modelling and analysis techniques. SMT-based analysis is proven to be an efficient back-end technology for formal analysis. We have gained insights on understanding WSN and their global behaviors. There are a number of case studies where SMT optimization techniques have shown their potentials.

Evaluation

During the course of our work, a few things have worked out nicely:

- A combination of techniques including reduction to SMT solving, Guarded Normal Form and multicore parallelism has made Duration Calculus much more compelling for practical use. The counting semantics technique has shown its usefulness in multiple situations.
- SMT optimization via νZ is a promising framework to leverage for analysis of CPS.
- The formal model is useful in reasoning about WSN. It lays the foundation for more rigorous analysis.

There are a number of things that can be improved. Duration Calculus is useful as a modelling formalism, but it is still not ready for large-scale use. Scalability of the approach used in Smart Meter use case is not yet well-understood. SMT optimization has huge potentials, but we have not fully exploited their capabilities in the case studies.

Directions for Future work

As future work, we would like to investigate along the following directions:

(A) The simulation-based approach in Paper D has shown interesting results on global behaviors of WSN. An attempt to analyze WSN in a more formal way

has been done for a chain topology in Chapter 4. An immediate direction is to extend the work for general topologies. Moreover, the WSN case study in Chapter 4 has been solved by an abstract semantics that provides templates for node operations. One can push the work further by doing local synthesis to obtain suitable parameters for efficient scheduling at each sensor node.

(B) The Smart Meter case study can be pushed further towards the use of a domain specific language. The formalization using Duration Calculus is nice but not intuitive enough for domain experts. One may derive a Duration Calculus variant catered for the domain or create a declarative domain specific language that employs Duration Calculus as an immediate representation.

(C) We have been using νZ for two simple case studies in the CPS area. While νZ has proven to be useful in such cases, there are other options to exploit SMT optimization for quantitative analysis. One could investigate more advanced case studies in CPS where νZ can provide powerful analysis backends, for example, by considering the extensions of (A) and (B) above.

References

[1] Rajeev Alur and David L. Dill. The theory of timed automata. In J. W. de Bakker, Cornelis Huizing, Willem P. de Roever, and Grzegorz Rozenberg, editors, *Real-Time: Theory in Practice, REX Workshop, Mook, The Netherlands, June 3-7, 1991, Proceedings*, volume 600 of *Lecture Notes in Computer Science*, pages 45–73. Springer, 1991.

[2] Gilles Audemard, Marco Bozzano, Alessandro Cimatti, and Roberto Sebastiani. Verifying industrial hybrid systems with mathsat. *Electronic Notes in Theoretical Computer Science*, 119(2):17–32, March 2005.

[3] Leo Bachmair, Ashish Tiwari, and Laurent Vigneron. Abstract congruence closure. *Journal of Automated Reasoning*, 31(2):129–168, 2003.

[4] Gerd Behrmann, Alexandre David, and Kim Guldstrand Larsen. A tutorial on uppaal. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT 2004, Bertinoro, Italy, September 13-18, 2004, Revised Lectures*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer, 2004.

[5] A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh. *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 2009.

[6] Armin Biere. Resolve and Expand. In Holger H. Hoos and David G. Mitchell, editors, *SAT (Selected Papers*, volume 3542 of *Lecture Notes in Computer Science*, pages 59–70. Springer, 2004.

[7] Nikolaj Bjørner. Linear Quantifier Elimination as an Abstract Decision Procedure. In *The 6th International Joint Conference on Automated Reasoning, IJCAR*, volume 6173 of *Lecture Notes in Computer Science*, pages 316–330. Springer, 2010.

[8] Guy E. Blelloch, Rezaul Alam Chowdhury, Phillip B. Gibbons, Vijaya Ramachandran, Shimin Chen, and Michael Kozuch. Provably good multicore cache performance for divide-and-conquer algorithms. In Shang-Hua Teng, editor, *Symposium on Discrete Algorithms*, pages 501–510. SIAM, 2008.

[9] Lei Bu, You Li, Linzhang Wang, and Xuandong Li. BACH : Bounded reachability checker for linear hybrid automata. In Alessandro Cimatti and Robert B. Jones, editors, *Formal Methods in Computer-Aided Design, FMCAD 2008, Portland, Oregon, USA, 17-20 November 2008*, pages 1–4. IEEE, 2008.

[10] Peter E. Bulychev, Alexandre David, Kim Guldstrand Larsen, Marius Mikucionis, Danny Bøgsted Poulsen, Axel Legay, and Zheng Wang. UPPAAL-SMC: statistical model checking for priced timed automata. In Herbert Wiklicky and Mieke Massink, editors, *Proceedings 10th Workshop on Quantitative Aspects of Programming Languages and Systems, QAPL 2012, Tallinn, Estonia, 31 March and 1 April 2012.*, volume 85 of *EPTCS*, pages 1–16, 2012.

[11] Zhou Chaochen and Michael R. Hansen. *Duration Calculus - A Formal Approach to Real-Time Systems.* Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[12] Zhou Chaochen, C. A. R. Hoare, and Anders P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.

[13] Alessandro Cimatti, Anders Franzén, Alberto Griggio, Roberto Sebastiani, and Cristian Stenico. Satisfiability modulo the theory of costs: Foundations and applications. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6015 of *Lecture Notes in Computer Science*, pages 99–113. Springer, 2010.

[14] Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. A Modular Approach to MaxSAT Modulo Theories. In Matti Järvisalo and Allen Van Gelder, editors, *Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings*, volume 7962 of *Lecture Notes in Computer Science*, pages 150–165. Springer, 2013.

[15] Edmund M. Clarke. The birth of model checking. In Orna Grumberg and Helmut Veith, editors, *25 Years of Model Checking - History, Achievements, Perspectives*, volume 5000 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2008.

[16] Sinem Coleri, Mustafa Ergen, and T. John Koo. Lifetime analysis of a sensor network with hybrid automata modelling. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, WSNA '02, pages 98– 104. ACM, 2002.

[17] Raffaello D'Andrea. Guest editorial can drones deliver? *IEEE Transactions on Automation Science and Engineering*, 11(3):647–648, 2014.

[18] Jessica Davies and Fahiem Bacchus. Postponing optimization to speed up MAXSAT solving. In Christian Schulte, editor, *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, volume 8124 of *Lecture Notes in Computer Science*, pages 247–262. Springer, 2013.

[19] Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[20] Conrado Daws, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. The tool KRONOS. In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Hybrid Systems III: Verification and Control, Proceedings of the DIMACS/SYCON Workshop, October 22-25, 1995, Ruttgers University, New Brunswick, NJ, USA*, volume 1066 of *Lecture Notes in Computer Science*, pages 208–219. Springer, 1995.

[21] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

[22] David Dharbe, Silvio Ranise, and Jorgiano Vidal. A prototype implementation of a distributed satisfiability modulo theories solver in the ToolBus framework. *Journal of the Brazilian Computer Society*, 14:71 – 86, 03 2008.

[23] Peter J. Downey, Ravi Sethi, and Robert Endre Tarjan. Variations on the common subexpression problem. *Journal of the ACM*, 27(4):758–771, October 1980.

[24] Bruno Dutertre and Leonardo Mendonça de Moura. A fast linear-arithmetic solver for DPLL(T). In Thomas Ball and Robert B. Jones, editors, *Computer Aided Verification, 18th International Conference, CAV 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, volume 4144 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2006.

[25] Peter H. Feiler, David P. Gluch, and John J. Hudak. The architecture analysis & design language (AADL): An introduction. Technical Report CMU/SEI-2006-TN-011, Software Engineering Institute, Carnegie Mellon University, 2006.

[26] Michael J. Fischer and Michael O. Rabin. Super-Exponential Complexity of Presburger Arithmetic. In *Proceedings of the SIAM-AMS Symposium in Applied Mathematics*, 1974.

[27] Martin Fränzle. Take it np-easy: Bounded model construction for duration calculus. In Werner Damm and Ernst-Rüdiger Olderog, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems, 7th International Symposium, FTRTFT 2002, Co-sponsored by IFIP WG 2.2, Oldenburg, Germany, September 9-12, 2002, Proceedings*, volume 2469 of *Lecture Notes in Computer Science*, pages 245–264. Springer, 2002.

[28] Martin Fränzle. Model-checking Dense-time Duration Calculus. *Formal Aspects of Computing*, 16(2):121–139, May 2004.

[29] Martin Fränzle and Michael R. Hansen. Deciding an Interval Logic with Accumulated Durations. In *13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, volume 4424 of *Lecture Notes in Computer Science*, pages 201–215. Springer, 2007.

[30] Martin Fränzle and Michael R. Hansen. Efficient Model Checking for Duration Calculus. *International Journal Software and Informatics*, 3(2-3):171–196, 2009.

[31] Peter Fritzson and Peter Bunus. Modelica-a general object-oriented language for continuous and discrete-event system modeling and simulation. In

Proceedings 35th Annual Simulation Symposium (ANSS-35 2002), San Diego, California, USA, 14-18 April 2002, pages 365–380. IEEE Computer Society, 2002.

[32] Sicun Gao, Jeremy Avigad, and Edmund M. Clarke. Delta-complete decision procedures for satisfiability over the reals. *ACM Computing Research Repository*, abs/1204.3513, 2012.

[33] Sicun Gao, Soonho Kong, Wei Chen, and Edmund M. Clarke. Delta-complete analysis for bounded reachability of hybrid systems. *ACM Computing Research Repository*, abs/1404.7171, 2014.

[34] Michael R. Hansen. Model-Checking Discrete Duration Calculus. *Formal Aspects of Computing*, 6(6A):826–845, 1994.

[35] Matthew Hause. The sysml modelling language. In *Fifteenth European Systems Engineering Conference*, 2006.

[36] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, LICS '96, pages 278–, Washington, DC, USA, 1996. IEEE Computer Society.

[37] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HYTECH: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1-2):110–122, 1997.

[38] Laura Humphrey, Eric Wolff, and Ufuk Topcu. Formal specification and synthesis of mission plans for unmanned aerial vehicles. In *AAAI Spring Symposium Series*, 2014.

[39] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1), February 2003.

[40] Mikkel Koefoed Jakobsen, Jan Madsen, and Michael R. Hansen. DEHAR: A distributed energy harvesting aware routing algorithm for ad-hoc multi-hop wireless sensor networks. In *Proceedings of the 2010 IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks*, WOWMOM '10, 2010.

[41] Richard E. Korf. A new algorithm for optimal bin packing. In Rina Dechter and Richard S. Sutton, editors, *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, July 28 - August 1, 2002, Edmonton, Alberta, Canada.*, pages 731–736. AAAI Press / The MIT Press, 2002.

[42] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1-2):134–152, 1997.

[43] Edward A. Lee and Sanjit A. Seshia. *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*. Lee and Seshia, 1 edition, 2010.

[44] Yi Li, Aws Albarghouthi, Zachary Kincaid, Arie Gurfinkel, and Marsha Chechik. Symbolic optimization with SMT solvers. In Suresh Jagannathan and Peter Sewell, editors, *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 607–618. ACM, 2014.

[45] David Monniaux. Quantifier Elimination by Lazy Model Enumeration. In Tayssir Touili, Byron Cook, and Paul Jackson, editors, *CAV*, volume 6174 of *Lecture Notes in Computer Science*, pages 585–599. Springer, 2010.

[46] António Morgado, Federico Heras, and João Marques-Silva. Improvements to Core-Guided Binary Search for MaxSAT. In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, volume 7317 of *Lecture Notes in Computer Science*, pages 284–297. Springer, 2012.

[47] B. Moszkowski. A Temporal Logic for Multilevel Reasoning about Hardware. *IEEE Computer*, 18(2):10–19, 1985.

[48] Shin Nakajima, Satoru Furukawa, and Yoshikazu Ueda. Co-analysis of sysml and simulink models for cyber-physical systems design. In *2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA 2012, Seoul, Korea (South), August 19-22, 2012*, pages 473–478. IEEE Computer Society, 2012.

[49] Nina Narodytska and Fahiem Bacchus. Maximum Satisfiability Using Core-Guided MaxSAT Resolution. In Carla E. Brodley and Peter Stone, editors, *AAAI*, pages 2717–2723. AAAI Press, 2014.

[50] Robert Nieuwenhuis and Albert Oliveras. On SAT Modulo Theories and Optimization Problems. In Armin Biere and Carla P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, volume 4121 of *Lecture Notes in Computer Science*, pages 156–169. Springer, 2006.

[51] Tobias Nipkow, Markus Wenzel, and Lawrence C. Paulson. *Isabelle/HOL: A Proof Assistant for Higher-order Logic*. Springer-Verlag, Berlin, Heidelberg, 2002.

[52] Peter Csaba Ölveczky and Stian Thorvaldsen. Formal modeling, performance estimation, and model checking of wireless sensor network algorithms in real-time maude. *Theoretical Computer Science*, 410(2-3):254–280, 2009.

[53] Derek C. Oppen. A $2^{2^{2^{pn}}}$ Upper Bound on the Complexity of Presburger Arithmetic. *J. Comput. Syst. Sci.*, 16(3):323–332, 1978.

[54] P.K. Pandya. Specifying and Deciding Quantified Discrete-time Duration Calculus Formulae using DCVALID. Technical Report TCS-00-PKP-1, Tata Institute of Fundamental Research, Mumbai, India, 2000.

[55] David A. Plaisted, Armin Biere, and Yunshan Zhu. A satisfiability procedure for quantified Boolean formulae. *Discrete Applied Mathematics*, 130(2):291–328, 2003.

[56] André Platzer. Differential dynamic logic for verifying parametric hybrid systems. In Nicola Olivetti, editor, *TABLEAUX*, volume 4548 of *LNCS*, pages 216–232. Springer, 2007.

[57] André Platzer and Edmund M. Clarke. Formal verification of curved flight collision avoidance maneuvers: A case study. In Ana Cavalcanti and Dennis Dams, editors, *FM*, volume 5850 of *LNCS*, pages 547–562. Springer, 2009.

[58] André Platzer and Jan-David Quesel. European Train Control System: A case study in formal verification. In Karin Breitman and Ana Cavalcanti, editors, *ICFEM*, volume 5885 of *LNCS*, pages 246–265. Springer, 2009.

[59] William Pugh. The Omega Test: a fast and practical integer programming algorithm for dependence analysis. In Joanne L. Martin, editor, *Supercomputing*, pages 4–13. IEEE Computer Society / ACM, 1991.

[60] Philip Reuterswärd, Johan Åkesson, Anton Cervin, and Karl-Erik Årzén. Truetime network—a network simulation library for modelica. In *International Modelica Conference*, 2009.

[61] Ludovic Samper, Florence Maraninchi, Laurent Mounier, and Louis Mandel. GLONEMO: global and accurate formal models for the analysis of ad-hoc sensor networks. In *Proceedings of the first international conference on Integrated internet ad hoc and sensor networks*, InterSense '06, 2006.

[62] Roberto Sebastiani and Silvia Tomasi. Optimization in SMT with $\mathcal{LA}(\mathbf{Q})$ Cost Functions. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, volume 7364 of *Lecture Notes in Computer Science*, pages 484–498. Springer, 2012.

[63] Conrado W. Seibel, Jean-Marie Farines, and José E. R. Cury. Towards using hybrid automata for the mission planning of unmanned aerial vehicles. In Panos J. Antsaklis, Wolf Kohn, Michael D. Lemmon, Anil Nerode, and Shankar Sastry, editors, *Hybrid Systems*, volume 1567 of *Lecture Notes in Computer Science*, pages 324–340. Springer, 1997.

[64] Helmut Seidl, Thomas Schwentick, Anca Muscholl, and Peter Habermehl. Counting in Trees for Free. In *31st International Colloquium on Automata, Languages and Programming, ICALP*, volume 3142 of *Lecture Notes in Computer Science*, pages 1136–1149. Springer, 2004.

[65] R C Shah and J M Rabaey. Energy Aware Routing for Low Energy Ad Hoc Sensor Networks. *IEEE Wireless Communications and Networking Conference Record*, 2002.

[66] N. Shankar, S. Owre, and J. M. Rushby. *PVS Tutorial*. Computer Science Laboratory, SRI International, Menlo Park, CA, February 1993. Also appears in Tutorial Notes, *Formal Methods Europe '93: Industrial-Strength Formal Methods*, pages 357–406, Odense, Denmark, April 1993.

[67] Babita Sharma, Paritosh K. Pandya, and Supratik Chakraborty. Bounded Validity Checking of Interval Duration Logic. In Nicolas Halbwachs and Lenore D. Zuck, editors, *11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, volume 3440 of *Lecture Notes in Computer Science*, pages 301–316. Springer, 2005.

[68] Paul Shaw. A constraint for bin packing. In Mark Wallace, editor, *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 of *Lecture Notes in Computer Science*, pages 648–662. Springer, 2004.

[69] Hossein M. Sheini, Bart Peintner, Karem A. Sakallah, and Martha E. Pollack. On solving soft temporal constraints using sat techniques. In Peter van Beek, editor, *CP*, volume 3709 of *Lecture Notes in Computer Science*, pages 607–621. Springer, 2005.

[70] Kin Cheong Sou, James Weimer, Henrik Sandberg, and Karl Henrik Johansson. Scheduling smart home appliances using mixed integer linear programming. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference, CDC-ECC 2011, Orlando, FL, USA, December 12-15, 2011*, pages 5144–5149. IEEE, 2011.

[71] Jean-Frédéric Étienne, S. Fechter, and E. Juppeaux. Using simulink design verifier for proving behavioral properties on a complex safety critical system in the ground transportation domain. In Marc Aiguier, Francis Bretaudeau, and Daniel Krob, editors, *CSDM*, pages 61–72. Springer, 2010.

[72] Ben L. Titzer, Daniel K. Lee, and Jens Palsberg. Avrora: scalable sensor network simulation with precise timing. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, IPSN '05, 2005.

[73] Sergio Yovine. KRONOS: A verification tool for real-time systems. *International Journal on Software Tools for Technology Transfer*, 1(1-2):123–133, 1997.

[74] Quan Zu, Miaomiao Zhang, Jiaqi Zhu, and Naijun Zhan. Bounded model-checking of discrete duration calculus. In Calin Belta and Franjo Ivancic, editors, *Proceedings of the 16th international conference on Hybrid systems: computation and control, HSCC 2013, April 8-11, 2013, Philadelphia, PA, USA*, pages 213–222. ACM, 2013.